Fountain Codes under Maximum Likelihood Decoding

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Francisco Lázaro Blasco

aus
Zaragoza

2017

Vorsitzender des Prüfungsausschusses:
Prof. Dr. Christian Schuster

1. Gutachter:
Prof. Dr. Gerhard Bauch

2. Gutachter:
Prof. Amin Shokrollahi

Tag der mündlichen Prüfung:
09.05.2017

To my beloved wife, parents and sister.

Acknowledgements

The journey towards this dissertation was fascinating but also long and challenging. It was only thanks to the technical advise, encouragement and love of those around me that I was able to finish it.

First, I would like to express my heartfelt gratitude to my advisor Prof. Gerhard Bauch for his guidance and valuable suggestions. I would also like to thank him explicitly for his support clearing the bureaucratic hurdles of the PhD. The next thank you goes to Prof. Amin Shokrollahi for accepting to review the thesis and for the helpful discussion during the PhD defense. I would also like to thank Prof. Christian Schuster for his efficiency handling the examination process.

Words cannot express my gratitude to my mentor at DLR, Gianluigi Liva, for his continuous support and guidance, and for teaching me all I know about channel coding. His passion and dedication to research have been an inspiration to me during all these years. I am also deeply in debt with Enrico Paolini for his advice and scientific rigour.

I would like to extend my gratitude also to my colleagues at DLR for creating a wonderful working atmosphere. Especially, I would like to thank Balázs, Federico, Giuliano and Javi for the technical and non-technical discussions. I would also like to thank Sandro Scalise and Simon Plass for supporting my research.

Last, but not least, I would like to thank my loving and caring family. I am infinitely thankful to my parents for giving me the most valuable present, a good education. A great thanks goes to my sister, for taking care of me like a mother. I would also like to thank my Italian family for making me feel welcome from the very first moment. I cannot put in words my gratitude to Paola for her love, patience, constant support and for making me feel at home. The very last “thank you” goes to little Francesco for pushing me to finalize this dissertation.

Munich, May 2017.

Abstract

This dissertation focuses on fountain codes under maximum likelihood (ML) decoding. Fountain codes are a class of erasure correcting codes that can generate an endless amount of coded symbols and were conceived to deliver data files over data networks to a potentially large population of users. First Luby transform (LT) codes are considered, which represent the first class of practical fountain codes. Concretely, the focus is on LT codes under inactivation decoding, an efficient ML decoding algorithm that is widely used in practical systems. More precisely, the decoding complexity of LT codes under inactivation decoding is analyzed in terms of the expected number of inactivations. The proposed analysis is based on a dynamical programming approach. This analysis is then extended to provide the probability distribution of the number of inactivations. Additionally a lower complexity approximate analysis is introduced and a code design example is presented that illustrates how these analysis techniques can be used to design LT codes. Next Raptor codes under ML decoding are considered. An upper bound to the probability of decoding failure of q-ary Raptor codes is developed, considering the weight enumerator of the outer code (precode). The bound is shown to be tight, specially in the error floor region, by means of simulations. This bound shows how Raptor codes can be analyzed similarly to a traditional serial concatenation of (fixed-rate) block codes. Next, a heuristic method is presented that yields an approximate analysis of Raptor codes under inactivation decoding. It is also shown by means of an example how the results in this thesis can be used to design Raptor codes. Raptor codes are next analyzed in a fixed-rate setting. Concretely, a Raptor code ensemble with an outer code picked from the linear random ensemble is considered. For this ensemble, the average weight enumerator and its growth rate are provided. Furthermore, sufficient and necessary conditions for the ensemble to have a minimum distance growing linearly with the block length are presented. The ensemble analyzed resembles standard Raptor codes, as it is shown by means of simulations. Finally a new class of fountain codes is introduced, that consists of a parallel concatenation of a block code with a linear random fountain code (LRFC). This scheme is specially

interesting when the block code is a maximum distance separable (MDS) code. In this case, the scheme can provide failure probabilities lower than those of LRFC codes by several orders of magnitude, provided that the erasure probability of the channel is not too high.

Table of contents

List of figures

List of tables

Acronyms

ARQ automatic retransmission query

AWGN additive white Gaussian noise

BCH Bose Chaudhuri Hocquenghem

BEC binary erasure channel

BSC binary symmetric channel

CER codeword error rate

CO-WEF conditional output-weight enumerator function

CRC cyclic redundancy check

FEC forward error correction

GE Gaussian elimination

GRS generalized Reed-Solomon

HDPC high-density parity-check

i.i.d. independent and identically distributed

IO-WEF input output-weight enumerator function

MDS maximum distance separable

ML maximum likelihood

LDPC low-density parity-check

LRFC linear random fountain code

Acronyms

LT Luby transform

QEC *q*-ary erasure channel

RS Reed-Solomon

RSD robust soliton distribution

SA simulated annealing

SPC single parity-check

WE weight enumerator

WEF weight enumerator function

Important Symbols

Roman Symbols

A_d	number of codewords of weight d
$\mathscr{C}$	code ensemble
h	number of intermediate symbols
H_b	binary entropy function
k	number of input symbols
m	number of output symbols collected by the receiver
n	number of output symbols generated by the encoder
P_F	probability of decoding failure
r	overall rate of a fixed-rate Raptor code
r_i	inner fixed-rate LT code rate
r_o	outer code rate
w_H	Hamming weight

Greek Symbols

Δ	Transmitter overhead
δ	absolute receiver overhead $\delta = m - k$
ϵ	relative receiver overhead, $\epsilon = m/k - 1$
ε	erasure probability of the channel

λ	normalized Hamming weight of the intermediate word $\lambda = l/h$
Ω	output degree distribution of an LT code
$\bar{\Omega}$	average output degree of an LT code
ϖ	normalized output weight $\varpi = w/n$

LT Codes: Vectors/Matrices

$\mathbf{c}$	row vector of output symbols
$\mathbf{G}$	generator matrix
$\tilde{\mathbf{G}}$	matrix corresponding to the non-erased positions of $\mathbf{G}$
$\mathbf{v}$	row vector of input (source) symbols
$\mathbf{y}$	row vector of received output symbols

Raptor codes: Vectors/Matrices

$\mathbf{c}$	row vector of output symbols
$\mathbf{G}_{\mathrm{LT}}$	generator matrix of the inner LT code
$\tilde{\mathbf{G}}_{\mathrm{LT}}$	matrix corresponding to the non-erased positions of $\mathbf{G}_{\mathrm{LT}}$
$\mathbf{H}_{\mathrm{p}}$	parity check matrix of the outer block code (precode)
$\mathbf{M}$	constraint matrix
$\mathbf{u}$	row vector of input (source) symbols
$\mathbf{v}$	row vector of intermediate symbols
$\mathbf{y}$	row vector of received output symbols

Other Symbols

$\mathbb{F}_q$	Galois field of order q
$\mathcal{K}_k(x; n, q)$	Krawtchouk polynomial of degree j with parameters h and q.
$\mathcal{N}$	neighbourhood of a node in a graph

$\mathcal{O}$	Landau big O notation
o	Landau small o notation
$\mathscr{P}$	positive growth rate region of a fixed-rate Raptor code
$\mathscr{R}$	ripple
$\mathscr{W}$	cloud

Operators

$\deg(c)$	degree of output symbol c
$\deg_r(c)$	reduced degree of output symbol c
$\lfloor x \rceil$	closest integer to x
$\binom{k}{i}$	binomial coefficient
$\lceil x \rceil$	smallest integer larger than or equal to x
$\lfloor x \rfloor$	largest integer smaller than or equal to x
$\operatorname{rank}(\mathbf{G})$	rank of matrix $\mathbf{G}$

Chapter 1

Introduction

> I just wondered how things were put together.
>
> Claude E. Shannon

In the early years of communication systems it was not known whether error free communication was possible over a communication channel that introduced errors using a rate that was not vanishingly small. It was C.E Shannon, who in his landmark paper from 1948 [1] proved that error free communication is possible if one communicates at a rate lower than the channel capacity. This milestone gave birth to the Information Age in which we live nowadays.

Initially the research community focused on the communication channels that arise in the physical layer of a communication system. At the physical layer of a communication system the thermal noise generated by thermal vibrations of the atoms in conductors can be accurately modeled as additive white Gaussian noise (AWGN), giving rise to the AWGN channel. The AWGN channel was one of the first models to be studied. Another simpler model of the physical layer is the binary symmetric channel (BSC) channel that was also widely studied during the early days of the Information Age. The BSC can be seen as a degradation of the AWGN when the input to the channel is constrained to be binary and symmetric and the receiver applies hard decision detection.

After the publication of Shannon's work a humongous amount of research has been carried out in the field of channel coding. The dominant motivation in the research community was getting closer and closer to Shannon's capacity with an affordable complexity. In the early decades of channel coding, algebraic codes were the main focus of research. The most prominent fruits of this research were Hamming, Golay, Reed Muller, BCH and RS codes [2–8]. Algebraic coding usually aims at finding codes with good distance properties, usually by maximizing the minimum distance of a code. Due to their good distance properties, algebraic codes tend to exhibit a low probability of error under optimal (maximum likelihood) decoding. The main disadvantage of algebraic codes, is that in general soft decoding tends to be complex, specially for large block lengths.

The first paradigm change in coding was shifting the focus towards *probabilistic codes* where the aim is at improving the average performance of a code with constraints on the encoding and decoding complexity [9]. At this stage, the research community had realized that the structure of the codes needed to be tailored to simplify the implementation in practical systems. Convolutional codes, introduced by Elias in [10] are generally considered to be the first class of probabilistic codes [9]. Optimal decoding algorithms for convolutional codes were first derived by Viterbi [11] and then by Bahl, Cocke, Jelinek and Raviv [12]. Another important milestone in coding was the introduction of concatenated codes by Forney [13], which involve a serial cascade of two linear block codes, usually denoted as inner and outer code. The main advantage of concatenated codes is that the inner and outer codes can be short and easy to decode. Hence, it is possible to decode concatenated codes using so called 2 stage decoders (decoding first the inner and then the outer coder). This decoder is suboptimal but it still shows a very good performance. In fact, the serial concatenation of RS and convolutional codes developed by NASA [14], and inspired in Forney's concatenated codes, was for many years one of the best performing coding schemes known and was widely used in practice.

The second paradigm change came with turbo codes, introduced in 1993 [15]. Thanks to iterative soft decoding algorithms both turbo and low-density parity-check (LDPC) codes were able to approach the Shannon limit in AWGN channels with a modest complexity. LDPC codes had been proposed and studied by Gallager in his doctoral thesis in 1963 [16] but later they had been largely forgotten because their potential for long block lengths was not recognised. Shortly after the introduction of turbo codes, LDPC codes were rediscovered in [17], where it was observed that their performance was better than that of convolutional and concatenated codes, and similar to that of turbo codes. Nowadays, the majority of practical wireless communication systems use turbo or LDPC codes since these codes allow to close largely the gap to capacity in most cases.

In the meantime digital communications have become ubiquitous and channel coding problems are no longer exclusive to the physical layer of communications systems. In this thesis we deal exclusively with erasure channels which are generally not typical from the physical layer. The binary erasure channel (BEC) was introduced by Elias in [10]. In this channel the transmitters sends one bit (either zero or one) and the receiver either receives this bit error free or receives an *erasure*. The BEC was originally regarded as a purely theoretical channel. However, this changed with the emergence of the Internet. It was soon realized that erasure channels are a very good abstraction model for the transmission of data over the Internet, where packets get lost due to, for example, buffer overflows at intermediate routers. Erasure channels also find applications in wireless and satellite channels where deep fading events can cause the loss of one or several packets.

Reliable communication in data networks can be achieved by using an automatic retransmission query (ARQ) mechanism in which the receiver requests the retransmissions of the information they have not been able to decode successfully. However, ARQ mechanisms present some limitations. The first is that they rely intensively on feedback. The second limitation enters into play in a reliable multicasting application, where one single transmitter wants to send an object (a file) to a set of receivers. In this scenario different receivers suffer different losses. If the number of receivers is large, the transmitter needs to process a large number of feedback messages and it also needs to perform a large number of retransmissions. For such applications, one would desire to have an alternative to ARQ that does not rely so much on feedback and whose efficiency scales better with the number of users.

Probably, one of the first works proposing erasure coding as an alternative to ARQ mechanisms is [18], where an algorithm is proposed for the transmission of a file to multiple receivers. Instead of retransmitting lost packets, the transmitter sends redundancy packets until all receivers acknowledge the reception of the file. In that work Reed-Solomon codes and linear random codes were considered, which become impractical due to their complexity for medium-large block lengths, i.e., for block lengths exceeding the few thousands.

Tornado codes were proposed for transmission over erasure channels [19, 20]. Tornado codes have linear encoding and decoding complexity (under iterative decoding). However, the encoding and decoding complexity is proportional to their block lengths and not their dimension, [19]. Hence, they are not suitable for low rate applications such as reliable multicasting, where the transmitter needs to adapt its code rate to the user with the worst channel (highest erasure probability). Another family of codes with good performance over erasure channels are LDPC codes. Several works have considered LDPC codes over erasure channels [21–23] and they have been proved to be practical in several scenarios even under maximum likelihood (ML) decoding. For example, in [23] a decoding speed of up to 1.5 Gbps was reported for a $(2048, 1024)$ LDPC using ML decoding. However, for a fixed code dimension, the decoding complexity of LDPC codes increases with the block length. Thus, as the erasure rate of the channel increases one is forced to increase the block length (i.e., decrease the rate), and the decoding complexity increases.

Although solutions based on linear block codes usually outperform ARQ mechanisms in the reliable multicasting setting, they still present some limitations. The first limitation is that the rate, and hence the block length, needs to be fixed a-priori. In the chosen rate turns out not to be low enough, it can happen that some users are unable to recover the original file. Furthermore, block codes usually need to be carefully designed taking into account the information and block lengths. Thus, if one decides to change these parameters one usually needs to carry out a new code design.

The concept of a digital fountain was introduced in [24] as an ideal solution to the problem of distributing data to a large number of users. A fountain code is basically an erasure code that is able to generate a potentially endless amount of encoded symbols. As such, fountain codes find application in contexts where the channel erasure rate is not known a priori. The first class of practical fountain codes, LT codes, was introduced in [25]. LT codes admit a sparse graph representation and can be decoded efficiently by means of iterative decoding when the code dimension (or number of input symbols, usually denoted by k) is large. The main drawback of LT codes is that in order to have a low probability of unsuccessful decoding, the encoding and iterative decoding cost per output/input[1] symbol has to grow at least logarithmically with the dimension of the code, k. Thus, LT codes have a scalability problem. On the one hand we need the number of input symbols k to be very large so that iterative decoding succeeds with high probability. On the other hand, by making k large the encoding and iterative decoding cost increase.

Raptor codes were introduced in [26] and published in [27],[28] as an evolution of LT codes. They were also independently proposed in [29], where they are referred to as online codes. Raptor codes consist of a serial concatenation of an outer block

[1] The encoding cost is defined as the encoding complexity in terms of operations normalized by the number of output symbols and the decoding cost as the decoding complexity normalized by the number of input symbols.

code, commonly referred to as precode, with an inner LT code. The basic idea behind Raptor codes is relaxing the LT code design, thus, requiring only the recovery of a fraction $1 - \gamma$ of the input symbols, where γ is usually small. This can be achieved with linear complexity, both in encoding and (iterative) decoding. The outer code is responsible for recovering the remaining fraction of input symbols, γ. If the precode is linear-time encodable, then the Raptor code has linear encoding complexity on the number of input symbols k, and therefore the overall encoding cost per output symbol is constant with respect to the number of input symbols k. If iterative decoding is used and the outer code can be decoded iteratively with linear complexity (in the number of input symbols k), the decoding complexity is also linear which results in a constant decoding cost per symbol. Furthermore, in [28] it was shown that Raptor codes under iterative decoding are universally capacity-achieving on the binary erasure channel. This means that a Raptor code can achieve the capacity of *all* BECs, no matter which value the erasure probability takes. Thus, they can be used for transmission over an erasure channel whose erasure probability is unknown and they are still guaranteed to achieve capacity.

Both LT and Raptor codes have been analyzed in depth under the assumption of iterative decoding and very large input blocks (at least in the order of a few tens of thousands symbols). However, often much smaller input block lengths are used due to different reasons. For example, the decoders have sometimes limited memory resources allocated, the files to be delivered are often of smaller size, and sometimes a short latency is desired. This leads to the need of efficient short fountain codes. This is the reason why, for the Raptor codes standardized in 3GPP Multimedia Broadcast Multicast Service (MBMS) and IETF it is recommend to use between 1024 and 8192 input symbols (see [30] and [31] for more details). For these input block lengths, the performance under iterative decoding degrades considerably. In fact, these codes are decoded using an efficient ML decoding algorithm known as inactivation decoding [32].

The focus of this doctoral thesis is on the analysis and design of fountain codes under ML decoding inspired by practical applications. Major parts of the results in this dissertation have been published in [33–41].

The remaining of this thesis is organized as follows. Chapter 2 provides some preliminaries on erasure channels, block codes and fountain codes. The two main classes of fountain codes, LT and Raptor codes are introduced in Chapter 3. In Chapter 4 LT codes under inactivation decoding are considered. The main contribution of this chapter is an analysis of the decoding complexity of LT codes under inactivation decoding using a dynamical programming approach. Chapter 5 focuses on Raptor codes under inactivation decoding. First, an upper bound on the probability of decoding failure of Raptor codes under ML decoding is presented. Then, a heuristic analysis of inactivation decoding is presented that provides an approximation of the number of inactivations. Chapter 6 contains several results related to the distance spectrum of an ensemble of fixed-rate Raptor codes. In Chapter 7 a novel fountain coding scheme is presented that consists of a parallel concatenation of a linear block code with a linear random fountain code (LRFC). This scheme is particularly interesting when the outer code is a maximum distance separable (MDS) code. Some concluding remarks are presented in Chapter 8. Appendix A contains a comparison of the performance of the different inactivation techniques used in practice. Finally, Appendix B contains some proofs that were omitted from Chapters 5 and 6.

Chapter 2

Background

> Everything should be made as simple as possible, but not simpler.
>
> Albert Einstein

In this chapter we briefly introduce the communication channels that are considered in this thesis. Concretely, we present three different channels, the binary erasure channel (BEC), the q-ary erasure channel (QEC) and the packet erasure channel. We then present some basic concepts related to block codes and fountain codes. Finally, the notation used in the thesis is described.

2.1 Channel Models

2.1.1 The Memoryless Binary Erasure Channel

The memoryless binary erasure channel (BEC) [10] is a communication channel with a binary input alphabet $\mathcal{X} = \{0, 1\}$ and a ternary output alphabet $\mathcal{Y} = \{0, 1, E\}$, as depicted in Figure 2.1. The symbol "E" denotes an erasure. Let $X \in \mathcal{X}$ be the random variable associated to the input of the channel and $Y \in \mathcal{Y}$ be the random variable associated with the output of the channel. The transition probabilities of the channel are:

$$\begin{aligned}
&\Pr(Y = y|X = x) = 1 - \varepsilon, \qquad \text{if } y = x,\\
&\Pr(Y = E|X = x) = \varepsilon.
\end{aligned}$$

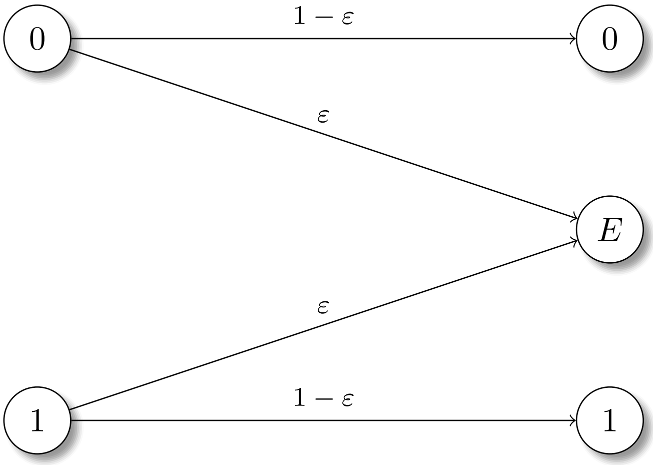


Fig. 2.1 The binary erasure channel (BEC).

When the symbols "0" or "1" are received there is no uncertainty about the symbol transmitted. However, when symbol "E" is received the receiver does not know which symbol was transmitted.

The capacity of the BEC is

$$C = 1 - \varepsilon \;\text{ [bits/channel use]},$$

and it is attained with X uniformly distributed.

2.1.2 The q-ary Erasure Channel

The q-ary erasure channel (QEC) is a communication channel with a q-ary input alphabet $\mathcal{X} = \{0, 1, ..., q-1\}$ and an output alphabet of cardinality $q+1$, $\mathcal{Y} = \{0, 1, ..., q-1, E\}$, as depicted in Figure 2.2. Again, symbol "E" denotes an erasure. Let $X \in \mathcal{X}$ be the random variable associated to the input of the channel and $Y \in \mathcal{Y}$ be the random variable associated to the output of the channel. The transition probabilities of the channel are:

$$\begin{aligned}\Pr(Y = y|X = x) &= 1 - \varepsilon, \qquad \text{if } y = x,\\ \Pr(Y = E|X = x) &= \varepsilon.\end{aligned}$$

The capacity of the QEC is

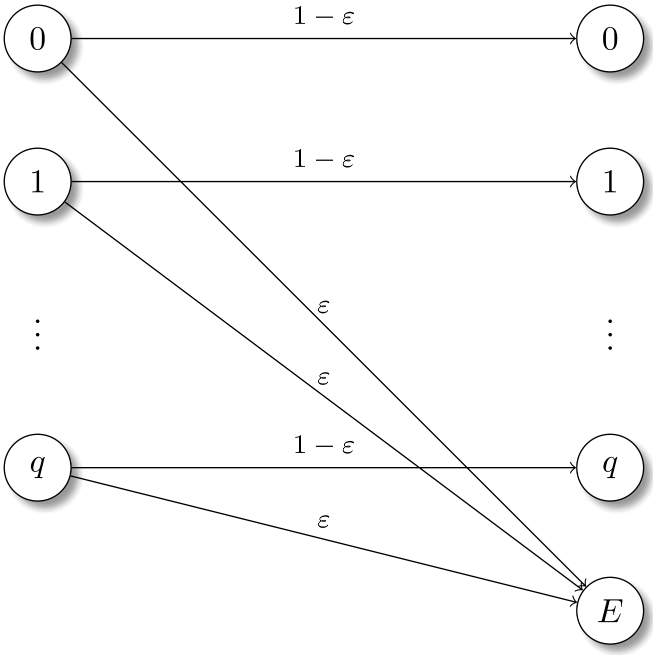


Fig. 2.2 The q-ary erasure channel (QEC).

$$C = 1 - \varepsilon \;\; \text{[symbols/channel use]},$$

and

$$C_b = \log_2(q)\, C \;\; \text{[bits/channel use]}.$$

The capacity is attained with X uniformly distributed in $\mathcal{X}$.

2.1.3 The Packet Erasure Channel

The packet erasure channel is a communication channel in which the input is a packet, that is, an array of L symbols belonging to the alphabet $\{0,1\}$, i.e. $\mathcal{X} = \{0,1\}^L$. Similarly to the BEC and QEC, in the packet erasure channel at the output the input is received error free with probability $1-\varepsilon$, and an erasure is received with probability ε.

The packet erasure channel can be seen as L parallel, fully correlated BECs [42]. Thus, the capacity of the packet erasure channel is

$$C = 1 - \varepsilon \;\; \text{[packets/channel use]},$$

and

$$C_b = LC \ \ \text{[bits/channel use]}.$$

Furthermore, all coding methods and performance bounds from the BEC can be applied to the packet erasure channel with slight modifications.

The packet erasure channel has a great practical importance. For example, let us consider a satellite or terrestrial communication link. The data to be transmitted is usually split into packets and each of these packets is transmitted using a channel code at the physical layer. At the receiver side, channel decoding is performed at the physical layer in order to correct the errors introduced by the (physical) channel. After channel decoding some residual errors might still be present. At this stage error detection is carried out and packets containing errors are marked as erased (discarded). It is easy to see how, under the assumption of perfect error detection, the upper layers can abstract the behavior of the lower layers as a packet erasure channel.

The packet erasure channel can also be used to abstract the behavior of a computer data network such as the Internet. In this case, generally, the packets need to be forwarded through different intermediate nodes before reaching their destination. In this case, packet losses can occur due to, for example, a buffer overflow in some intermediate node. Additionally, during transmission bit errors can occur. Protocols (i.e. IP protocol) usually add a cyclic redundancy check (CRC) to each packet, that is used to detect and discard erroneous packets. All in all, the behavior of the data network can be abstracted by the upper layers as a packet erasure channel between the encoder and decoder.

Figure 2.3 shows the block diagram of a typical digital communication system that makes use of erasure coding in a single link communication. At upper layers, a packet erasure channel encoder is used which accepts at its input k source packets and generates n output packets. Before transmission, each frame is protected by an erasure code. At the receiver side channel decoding is performed at the physical layer in order to correct the errors introduced by the (physical layer) channel. After channel decoding some residual errors might be present. At this stage error detection is carried out and packets containing errors are marked as erased (discarded). Next, this packets are passed on to the packet erasure channel decoder which then recovers the k original source packets.

Due to the easy mapping of the packet erasure channel to the BEC and QEC, for ease of exposition all the results in this thesis will be stated in the BEC/QEC setting,

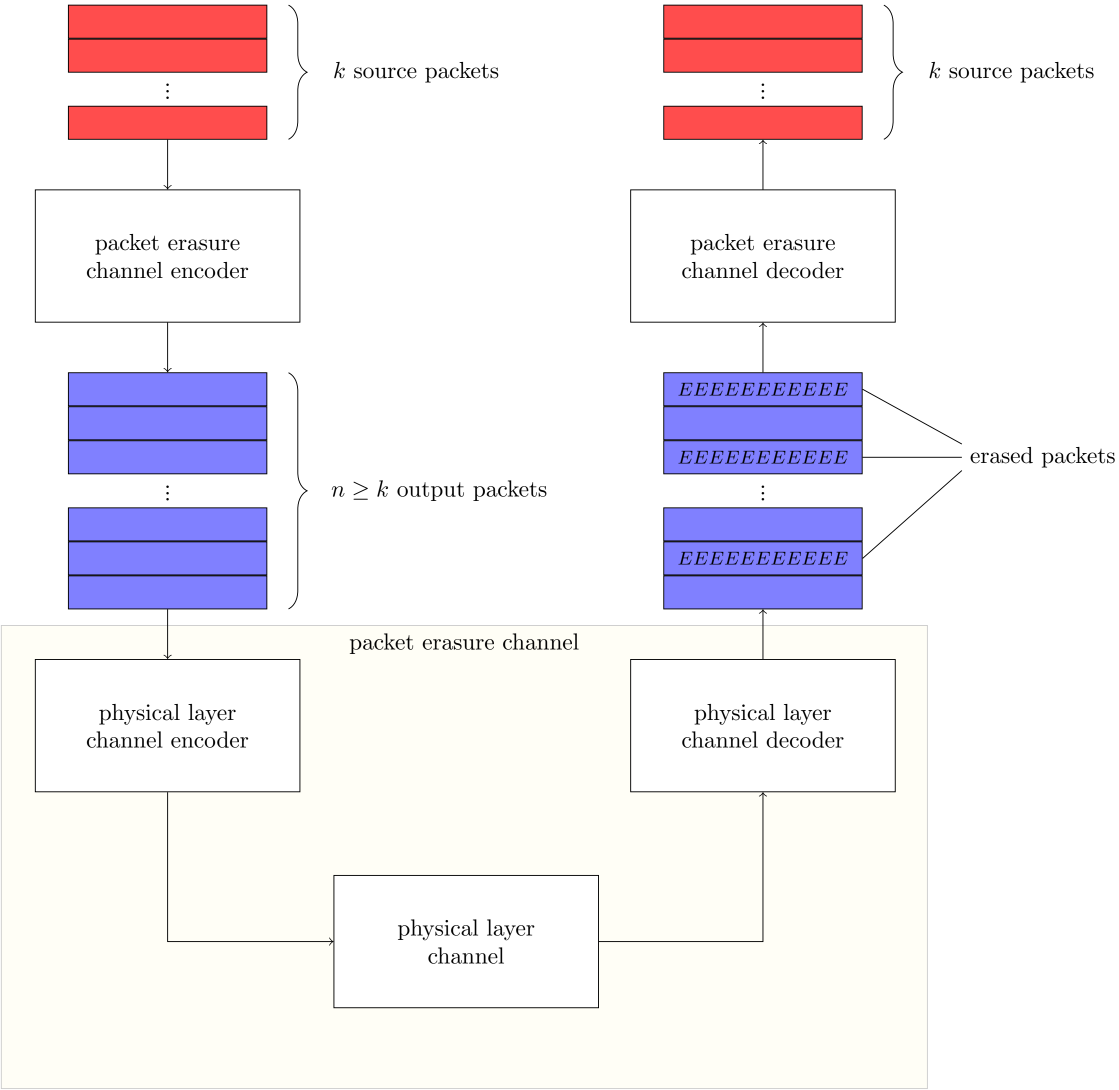


Fig. 2.3 Simplified diagram of a communication system that makes use of packet erasure coding.

being the extension to the packet erasure channel straightforward. This approach is quite widespread in the recent literature of coding for erasure channels.

2.2 Block Codes: Basics and Performance Bounds

Consider the transmission over the BEC with a (n, k) binary linear block code $\mathcal{C}$. It is possible to show that the block error probability, P_B satisfies the following inequality

$$P_B(\mathcal{C}) \geq P_B^{(\mathsf{S})},$$

where $P_B^{(\mathsf{S})}$ is the Singleton bound [43],

$$P_B^{(\mathsf{S})}(n, k, \varepsilon) = \sum_{e=n-k+1}^{n} \binom{n}{e} \varepsilon^e (1-\varepsilon)^{n-e}. \tag{2.1}$$

In this bound, equality is achieved only if $\mathcal{C}$ is a (n, k) maximum distance separable (MDS) code, i.e., if the code minimum distance is:

$$d_{\min} = n - k + 1.$$

Berlekamp derived an upper bound on the *average* block error probability of random binary linear block codes [44]

$$\begin{aligned} P_B^{(\mathsf{B})} = & \sum_{e=n-k+1}^{n} \binom{n}{e} \varepsilon^e (1-\varepsilon)^{n-e} \\ & + \sum_{e=1}^{n-k} \binom{n}{e} \varepsilon^e (1-\varepsilon)^{n-e} \, 2^{-(n-k-e)}. \end{aligned} \tag{2.2}$$

If we compare (2.1) and (2.2) we can see how the Berlekamp bound is composed of the Singleton bound plus a correction term.

Let us denote by $\mathcal{C}^*$ the best code among all (n, k) binary linear block codes, where by best we mean the one with the minimum block error probability over a BEC. We have that:

$$P_B^{(\mathsf{S})} \leq P_B(\mathcal{C}^*) < P_B^{(\mathsf{B})}.$$

That is, the Singleton and the Berlekamp bounds provide lower and upper bounds to the block error probability of the best binary linear block code with parameters (n, k).

The block error probability of a linear block code not only depends on its minimum distance, $d_{\min}$, but also on its weight enumerator, A_w, that corresponds to the number of codewords of Hamming weight w. Unfortunately, when dealing with modern (turbo/ LDPC) codes, deriving the exact weight enumerator of a code is a very challenging problem [45]. For this reason it is convenient to work with code ensembles since it is usually easier to derive average results for the ensemble.

A code ensemble $\mathscr{C}$ is a set of codes $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$ together with a probability distribution that gives the probability of the occurrence of each of the codes in the ensemble. We will illustrate the concept of code ensemble by means of an example.

Example 1. *The (n,k) binary linear random ensemble is given by all possible codes obtained by generating at random a $(n-k) \times n$ parity check matrix $\mathbf{H}$ in which each element of the parity check matrix takes value one with probability 1/2. This ensemble contains all (n,k') codes with $k' \leq k$, since the rank of $\mathbf{H}$ can be smaller than $n-k$.*

Let us consider a binary linear block code ensemble $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$. The ensemble average weight enumerator $\mathcal{A}_w$ is defined as

$$\mathcal{A}_w = \mathbb{E}_{\mathcal{C} \in \mathscr{C}} \left[A_w(\mathcal{C}) \right],$$

where $\mathbb{E}_{\mathcal{C} \in \mathscr{C}}[\cdot]$ denotes expectation over all the codes $\mathcal{C}$ in the ensemble $\mathscr{C}$, and $A_w(\mathcal{C})$ is the weight enumerator of code $\mathcal{C}$.

Consider a binary linear block code ensemble $\mathscr{C}$ with average weight enumerator $\mathcal{A}_w$. The average block error probability for codes in the ensemble, $P_B(\mathscr{C})$, can be upper bounded as [46]

$$\begin{aligned} P_B(\mathscr{C}) = \mathbb{E}_{\mathcal{C} \in \mathscr{C}} \left[P_B(\mathcal{C}) \right] \leq P_B^{(\mathsf{S})}(n,k,\varepsilon) \\ + \sum_{e=1}^{n-k} \binom{n}{e} \varepsilon^e (1-\varepsilon)^{n-e} \min \left\{ 1, \sum_{w=1}^{e} \binom{e}{w} \frac{\mathcal{A}_w}{\binom{n}{w}} \right\}. \end{aligned} \tag{2.3}$$

2.3 Fountain Codes: Basics and Performance Bounds

Consider a fountain code $\mathcal{C}_f$ of dimension k. The fountain encoder receives at its input k input symbols (also called source symbols) out of which it generates n output symbols (also called coded symbols). The key property of a fountain code is that the number of output symbols n does not need to be fixed a-priori. Additional output symbols can be generated *on the fly* in an on-demand fashion. For this reason, fountain codes are said to be rateless.

We consider the transmission over an erasure channel with a fountain code with k input symbols. In this setting, the output symbols generated by the fountain encoder are transmitted through an erasure channel where they are erased with probability ε.

We denote by m the number of output symbols that are not erased by the channel at a given receiver. We define the absolute (receiver) overhead as:

$$\delta := m - k.$$

We also define the relative overhead ϵ as the absolute overhead normalized by the number of input symbols, formally:

$$\epsilon := \frac{\delta}{k} = m/k - 1.$$

Given the fact that fountain codes are rateless (n not fixed) it is useful to define the performance bounds of fountain codes in terms of the absolute receiver overhead. More concretely, we are interested in bounds to the probability of decoding failure as a function of the absolute receiver overhead, $\mathsf{P_F}(\delta)$.

A lower bound to the performance of fountain codes is obtained assuming an *ideal* fountain code that allows the receiver to decode successfully whenever $m \geq k$ output symbols are received, i.e., whenever $\delta \geq 0$. The performance on an ideal fountain code is, hence, given by:

$$\mathsf{P}_\mathsf{F}^\mathrm{I}(\delta) = \begin{cases} 1 & \delta < 0 \\ 0 & \delta \geq 0 \end{cases}.$$

Thus, for any given fountain code $\mathcal{C}_f$ its decoding failure probability can be lower bounded as

$$\mathsf{P_F}(\mathcal{C}_f, \delta) \geq \mathsf{P}_\mathsf{F}^\mathrm{I}(\delta)$$

Let us consider a linear random fountain code (LRFC)[1] on a finite field of order q. In [47] it was shown how the probability of decoding failure of an LRFC can be upper bounded as

$$\mathsf{P_F}(\delta) < \frac{1}{q-1} q^{-\delta}, \qquad \delta \geq 0 \tag{2.4}$$

Let us now denote by $\mathcal{C}_f^*$ the best code among all q-ary fountain codes with k input symbols, where by best we mean the one with the minimum block error probability over a QEC. We have that:

$$\mathsf{P}_\mathsf{F}^\mathrm{I}(\delta) \leq \mathsf{P_F}(\mathcal{C}_f^*, \delta) < \frac{1}{q-1} q^{-\delta}, \qquad \delta \geq 0$$

[1] linear random fountain codes (LRFCs) are defined in Section 3.1.

That is, the performance of an ideal fountain code and the bound in (2.4) provide lower and upper bounds to the probability of decoding failure of the best q-ary fountain code with k input symbols, when used to transmit over a q-ary erasure channel.

2.4 Notation

In this section we introduce several definitions which will be used throughout the thesis.

Definition 1 ($\mathcal{O}$-notation)**.** *Let f and g be two real functions. We write:*

$$f(n) = \mathcal{O}\left(g(n)\right), \quad \textit{as } n \to \infty,$$

if for sufficiently large values of n, there exists a constant s so that

$$|f(n)| \leq s|g(n)|.$$

For example, if a function f is $\mathcal{O}(\log(n))$, given n, we can find a value s such that f is upper bounded by $s\log(n)$ for sufficiently large n. This notation is also known as Landau notation and it is employed to characterize the behaviour of a function when its argument tends to infinity [48].

Another useful asymptotic notation is the small o-notation whose formal definition is introduced next.

Definition 2 (o-notation)**.** *Let f and g be two real functions. We write:*

$$f(n) = \mathrm{o}\left(g(n)\right), \quad \textit{as } n \to \infty,$$

if and only if for any constant $s > 0$ and sufficiently large n

$$|f(n)| \leq s|g(n)|.$$

Note that although the definitions of $\mathcal{O}$-notation and o-notation are similar, they are not equivalent. For example, consider $f(n) = n^2$. We can say that n^2 is $\mathcal{O}(n^2)$ but this would not be true for little o-notation.

Definition 3 (Exponential equivalence)**.** *Two real-valued positive sequences $a(n)$ and $b(n)$ are said to be exponentially equivalent [49], writing $a(n) \doteq b(n)$, when*

$$\lim_{n\to\infty} \frac{1}{n} \log_2 \frac{a(n)}{b(n)} = 0.$$

If $a(n)$ and $b(n)$ are exponentially equivalent, then

$$\lim_{n\to\infty} \frac{1}{n} \log_2 a(n) = \lim_{n\to\infty} \frac{1}{n} \log_2 b(n).$$

Chapter 3

Linear Random Fountain Codes, LT and Raptor Codes

Within this chapter we present three fountain code constructions that can be found in literature. First we introduce linear random fountain codes (LRFCs), which are probably the conceptually simplest fountain code one can think of. We then introduce LT codes, and describe their encoding and decoding procedures. Finally, we introduce Raptor codes, which are arguably the best performing fountain coding scheme known.

3.1 Linear Random Fountain Codes

For the sake of completeness, let us start by formally defining a Galois Field

Definition 4 (Galois Field)**.** *We denote by $\mathbb{F}_q$ a Galois field or finite field of order q. A Galois Field $\mathbb{F}_q$ is a set of q elements on which the addition and multiplication operations fulfil the following properties:*

(a) $\mathbb{F}_q$ is an Abelian group under addition with identity element denoted by 0.

(b) $\mathbb{F}_q \backslash \{0\}$ is a multiplicative group with identity element denoted by 1.

(c) multiplication is distributive over addition

A q-ary linear random fountain code (LRFC) is a fountain code that accepts at its input a set of k input (or source) symbols, $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_k)$, where $v_i \in \mathbb{F}_q$. At its output, the linear random fountain code encoder can generate an unlimited amount of output symbols (also known as coded symbols) $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, where n can grow indefinitely and $c_i \in \mathbb{F}_q$. The i-th output symbol c_i is generated as:

$$c_i = \sum_{j=1}^{k} g_{j,i} v_j,$$

where the coefficients $g_{j,i}$ are picked from $\mathbb{F}_q$ with uniform probability. If we assume n to be fixed, LRFC encoding can be seen as a vector matrix multiplication:

$$\mathbf{c} = \mathbf{v}\mathbf{G},$$

where $\mathbf{G}$ is an $k \times n$ with elements $g_{j,i}$ picked uniformly at random from $\mathbb{F}_q$.

Let us now assume that the output symbols produced by the LRFC encoder are transmitted over a q-ary erasure channel, and let us also assume that out of the n output symbols generated by the LRFC encoder, the receiver collects $m = k + \delta$, denoted by $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. Denoting by $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ the set of indices corresponding to the m non-erased symbols, we have

$$y_j = c_{i_j}.$$

We can now cast the received output symbols as

$$\mathbf{y} = \mathbf{v}\tilde{\mathbf{G}} \tag{3.1}$$

with $\tilde{\mathbf{G}}$ given by the m columns of $\mathbf{G}$ with indices in $\mathcal{I}$.

LRFC decoding is performed by solving the system of equations in (3.1). Note that matrix $\tilde{\mathbf{G}}$ is dense, since its elements are picked uniformly at random in $\mathbb{F}_q$. Due to the high density of $\tilde{\mathbf{G}}$ LRFC decoding is quite complex; hence, LRFCs are only practical for small values of k (at most in the order of the hundreds).

The performance of these codes is remarkably good and follows a relatively simple model. Under ML decoding, the decoding failure probability of a binary LRFC [28, 50] can be accurately modeled as $\mathsf{P_F} \sim 2^{-\delta}$ for $\delta \geq 0$. Actually, $\mathsf{P_F}$ can be upper bounded by $2^{-\delta}$ [44, 28, 50].

In [51], LRFC on finite fields of order equal or larger than 2 ($\mathbb{F}_q$, $q \geq 2$) were analyzed. It was shown that for an LRFC over $\mathbb{F}_q$, the failure probability under ML decoding is bounded as [51]

$$q^{-\delta-1} \leq \mathsf{P_F}(\delta, q) < \frac{1}{q-1} q^{-\delta} \tag{3.2}$$

where both bounds are tight already for $q = 2$, and become tighter for increasing q.

3.2 LT codes

Luby transform (LT) codes were introduced in [25] as the first practical implementation of a fountain code. They were originally introduced together with an iterative decoding algorithm that will be explained in detail in Section 3.2.1.

An LT code accepts at its input a set of k symbols, $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_k)$, that are commonly referred to as input symbols (or source) symbols. At its output, the LT encoder can generate an unlimited amount of output symbols (also known as coded symbols) $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, where n can grow indefinitely. A key concept when dealing with LT codes is the degree of an output symbol or output degree, which is defined as the number of input symbols that were used to generate the output symbol under consideration. An LT code is defined by an output degree distribution $\Omega = (\Omega_1, \Omega_2, \Omega_3, \ldots, \Omega_{d_{\max}})$, where Ω_d corresponds to the probability that an output symbol of degree d is generated, and $d_{\max}$ is the maximum output degree.

In order to generate one output symbol the LT encoder performs the following steps:

- Randomly choose a degree d according to the degree distribution Ω.
- Choose uniformly at random d distinct input symbols.
- Compute the output symbol as a xor of the d selected input symbols.

If we assume for a moment that the number of output symbols n is fixed, the LT encoding operation can be seen as a vector matrix multiplication:

$$\mathbf{c} = \mathbf{v}\mathbf{G},$$

where $\mathbf{G}$ is an $k \times n$ binary[1] matrix which defines the relation between the input and the output symbols. The element $g_{i,j}$ of $\mathbf{G}$ is set to one only if input symbol i was

[1] Unless otherwise stated we will always consider binary LT codes.

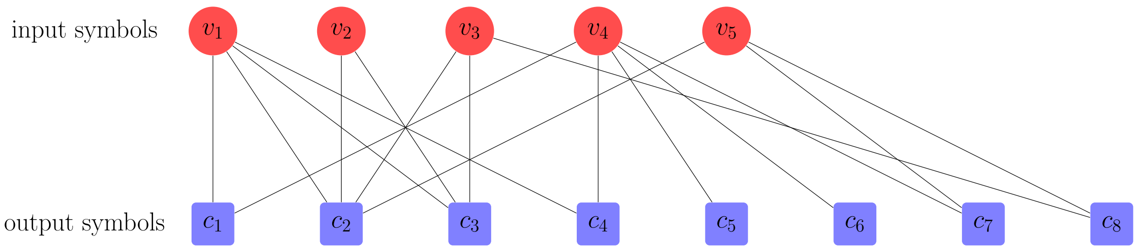


Fig. 3.1 Bipartite graph of an LT code.

used to generate output symbol j. Otherwise, element $g_{i,j}$ is set to zero. From this description it is easy to see how binary LRFCs can be considered a particular type of LT code in which the output degree distribution corresponds to a binomial distribution with parameters k and $1/2$.

LT codes admit a bipartite graph representation. In the bipartite graph of an LT code there are two different types of nodes, corresponding to input and output symbols. Let us introduce the notation $\deg(c)$ to refer to the degree of an output symbol c. An output symbol of degree d will have d neighbors in the bipartite graph. We will use the notation $\mathcal{N}(\cdot)$ to denote the set of neighbours, i.e. the neighbourhood of a node.

The bipartite graph of an LT code is related to its matrix representation, and can be derived from $\mathbf{G}$. We will illustrate this by means of an example. Figure 3.1 shows the bipartite graph representation of an LT code with $k = 5$ input symbols and $n = 8$ output symbols. In the figure, input symbol are represented by red circles and output symbol using blue squares. The generator matrix of the LT code represented in the figure corresponds to

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

An important parameter of an LT code is its average output degree $\bar{\Omega}$, that is given by

$$\bar{\Omega} := \sum_{i=1}^{d_{\max}} i\Omega_i.$$

In LT literature, degree distributions are commonly represented in polynomial form. Given a degree distribution Ω, its polynomial representation $\Omega(\mathtt{x})$ is given by

$$\Omega(\mathtt{x}) := \sum_{i=1}^{d_{\max}} \Omega_i \mathtt{x}^i.$$

This representation can be used to derive moments of the degree distribution (that is a probability mass function) in a very compact form. For example, the average output degree can be expressed as the first derivative of $\Omega(\mathtt{x})$ evaluated at 1,

$$\bar{\Omega} = \Omega'(1).$$

3.2.1 Iterative Decoding

LT codes were introduced in [25] together with a suboptimal, low complexity decoding algorithm. Although a more proper name for it would be that of peeling decoder, this decoder is usually referred to as iterative decoder. In this thesis we will use the terms iterative decoding and peeling decoder interchangeably.

Iterative decoding of LT codes is best described using a bipartite graph. Let us assume that the receiver has collected $m = k + \delta$ output symbols that we will denote by $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. We will consider a bipartite graph containing the m collected output symbols, $\mathbf{y}$, and the k input symbols $\mathbf{v}$.

Algorithm 1 (Iterative decoding)**.**

1. *Search for an output symbol of degree one.*
    (a) *If such an output symbol y exists move to step 2.*
    (b) *If no output symbols of degree one exist, iterative decoding exits and decoding fails.*
2. *Output symbol y has degree one. Thus, denoting its only neighbour as v, the value of v is recovered by setting $v = y$.*
3. *Denoting by $\mathcal{N}(v)$ the set of neighbours of v. For each $y \in \mathcal{N}(v)$:*
    (a) *Update the value of y as: $y = y + v$, where $+$ denotes addition over $\mathbb{F}_2$.*
    (b) *Remove input symbol v and all its attached edges from the graph.*

4. If all k input symbols have been recovered, decoding is successful and iterative decoding ends. Otherwise, go to step 1.

In order to illustrate iterative decoding we will provide a small example. Figure 3.2 shows the bipartite graph before iterative decoding starts. We can see that the number of source symbols is $k = 4$ and the number of output symbols collected by the receiver (not erased by the channel) is $n = 5$.

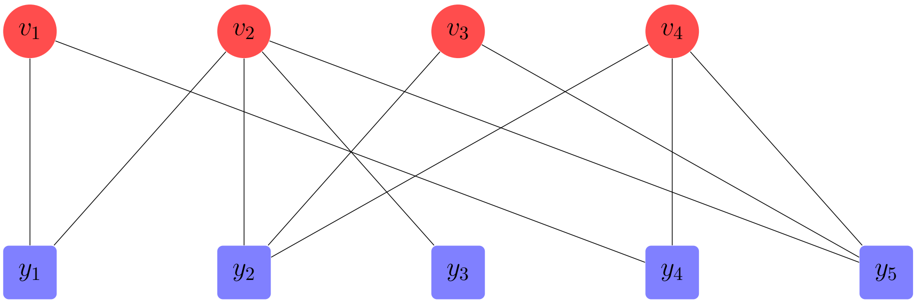


Fig. 3.2 Iterative decoding example, step 0.

Iterative decoding starts by searching for a degree one output symbol. In Figure 3.3 we can see that output symbol y_3 is the only output symbol with degree one. Using y_3 the decoder recovers v_2. Afterwards, the decoder performs the xor (addition over $\mathbb{F}_2$) of v_2 with all its neighbors. After doing so all edges attached to v_2 are erased.

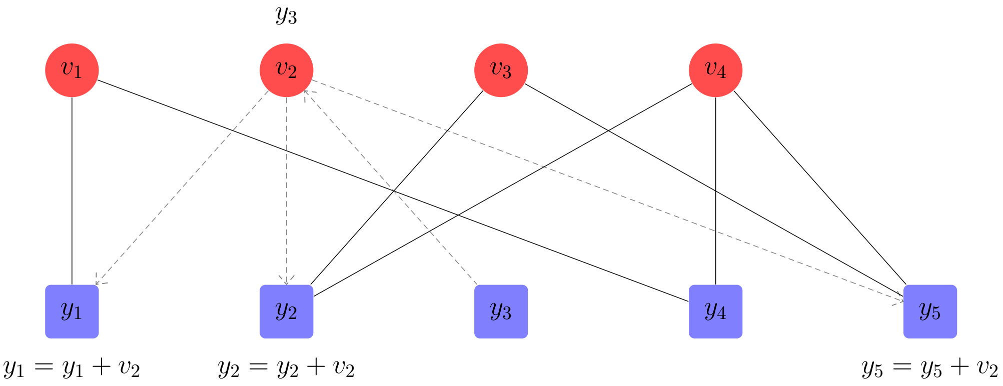


Fig. 3.3 Iterative decoding example, step 1.

The second run of iterative decoding is shown in Figure 3.4. The decoder finds the only degree one output symbol y_1, and uses it to recover v_1. Next, the decoder performs the xor (addition over $\mathbb{F}_2$) of v_1 with its other neighbor, y_4, and erases the edges attached to v_1.

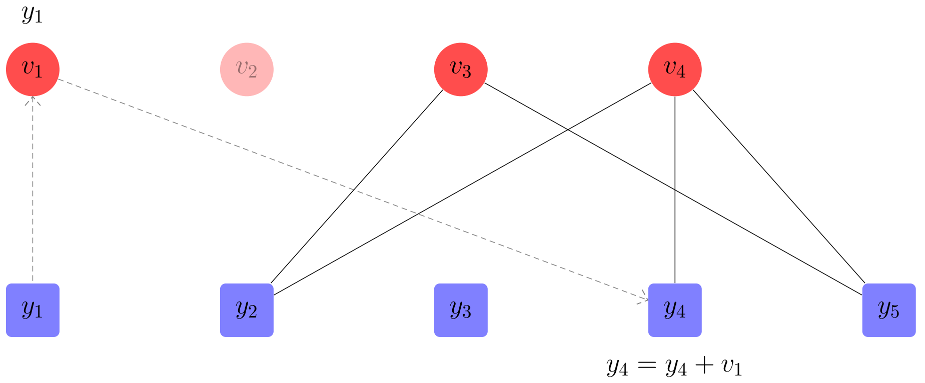


Fig. 3.4 Iterative decoding example, step 2.

Figure 3.5 depicts the third iteration. We can see how the only degree one output symbol y_4 is used to solve v_4. Then the decoder performs the xor of v_4 to its other neighbor, y_2 and the edges are removed from the graph.

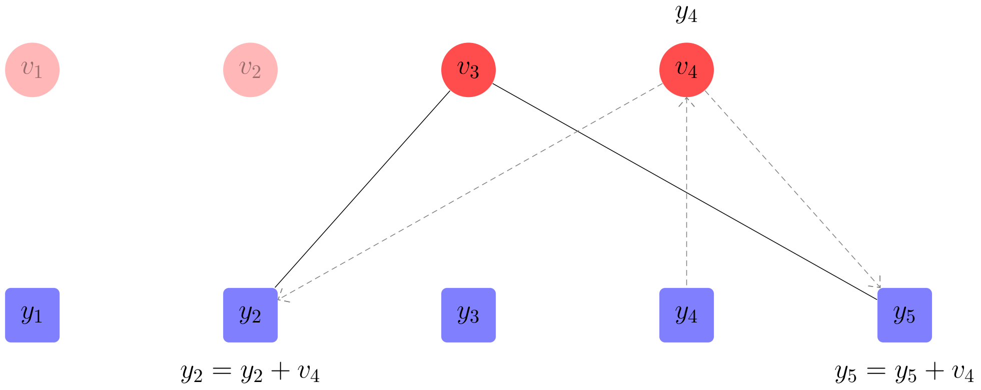


Fig. 3.5 Iterative decoding example, step 3.

Finally, the last iteration is shown in Figure 3.6. Now there are two degree one output symbols, y_2 and y_5. In this case we assume the decoder chooses at random y_2 to recover the last input symbol v_3.

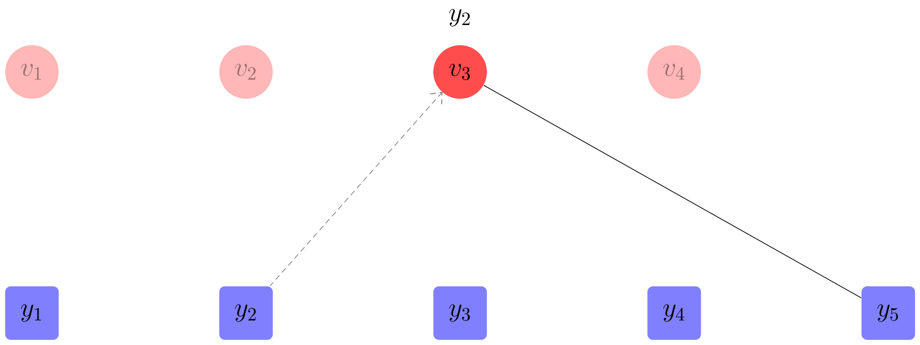


Fig. 3.6 Iterative decoding example, step 4.

The following proposition ([25]) provides a necessary condition for decoding to be successful with high probability.

Proposition 1. *A necessary condition for decoding to be successful with high probability is* $\bar{\Omega} = \mathcal{O}\left(\log(k)\right)$.

Proof. The proof uses the "balls into bins" argument that was presented in [25]. Let us first assume that k and m are very large and let us assume that at encoding each output symbol chooses its neighbors with replacement[2]. Let us consider a randomly chosen input symbol v and an output symbol y of degree d. The probability that v is not in the neighborhood of y corresponds to:

$$\Pr\{v \notin \mathcal{N}\left(y\right) | \deg(y) = d\} = \left(\frac{k-1}{k}\right)^d .$$

Let us denote by P_e the probability that v does not have any edges to the m received symbols. This probability corresponds to the probability of v not belonging to the union of the neighborhoods of the m received output symbols. Under the replacement assumption we have that

$$P_e = \Pr\left\{v \notin \bigcup_{i=1}^{m} \mathcal{N}\left(y_i\right) \Big| \sum_{i=1}^{m} \deg(y_i)\right\} = \left(\frac{k-1}{k}\right)^{\sum_{i=1}^{m} \deg(y_i)}$$

[2] This means that an output symbol will be allowed to choose multiple times the same neighbor. However, this will happen with a negligible probability for large enough values of k.

If we now let k tend to infinity, we have

$$\lim_{k\to\infty} P_e = e^{-\bar{\Omega}(1+\epsilon)}$$

where we have made use of the relationship

$$\lim_{k\to\infty} \left(\frac{k-1}{k}\right)^k = e^{-1}.$$

Let us denote by ϕ the expected number of input symbols not covered by any output symbol,

$$\phi = ke^{-\bar{\Omega}(1+\epsilon)}.$$

A necessary condition for successful decoding with high probability is that the ϕ is vanishingly small. If we relax this condition and let ϕ simply be a small positive number, we have

$$\bar{\Omega} = \frac{\log(k/\phi)}{1+\epsilon}.$$

This leads us to the statement in the proposition. □

Note that the condition in Proposition 1 is valid for any decoding algorithm and not only for iterative decoding.

The performance of LT codes under iterative decoding has been object of study in several works and is well understood, [52–55]. Iterative decoding of LT codes can be seen as an iterative pruning of the bipartite graph of the LT code. If we take an instance of decoding in which iterative decoding is successful, we have that initially all input symbols are unresolved (not yet decoded). At every iteration exactly one input symbol is resolved and all edges attached to the resolved input symbol are erased from the graph. Decoding continues until all input symbols are resolved, which is the case after k iterations. Let us consider the iterative decoder at some intermediate step in which u input symbols are yet unresolved and $k-u$ symbols have already been resolved. Following [52] we shall introduce some definitions that provide an insight into the iterative decoding process.

Definition 5 (Reduced degree). *We define the reduced degree of an output symbol as the degree of the output symbol in a reduced bipartite graph in which only unresolved input symbols are present.*

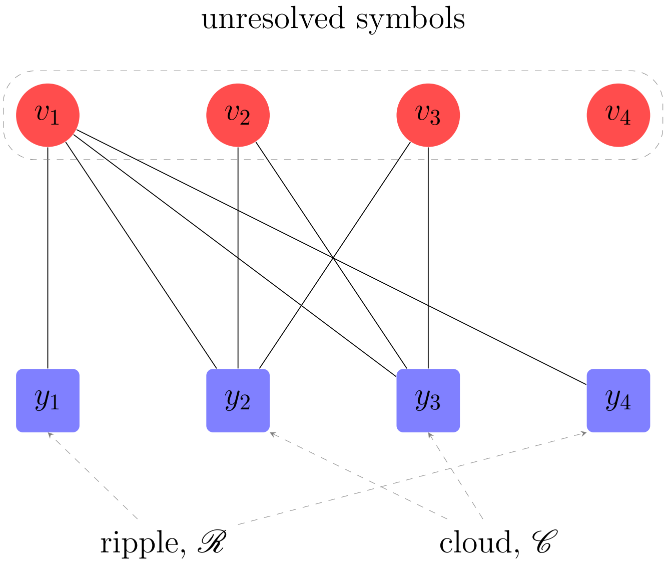


Fig. 3.7 Example of ripple and cloud in the bipartite graph of an LT code.

Thus, at the initial stage of iterative decoding, when all input symbols are unresolved, the reduced degree of a symbol is equal to its actual degree. However, as iterative decoding progresses the reduced degree of an output symbol decreases if his neighbors get resolved.

Definition 6 (Output ripple)**.** *We define the output ripple or simply ripple as the set of output symbols of reduced degree 1 and we denote it by* $\mathscr{R}$*.*

Definition 7 (Cloud)**.** *We define the cloud as the set of output symbols of reduced degree* $d \geq 2$ *and we denote it by* $\mathscr{W}$*.*

Figure 3.7 shows the bipartite graph of an LT code in which 4 input symbols are unresolved. It can be observed how output symbols y_1 and y_4 belong to the ripple since they have reduced degree one and output symbols y_2 and y_3 belong to the cloud since their degree is 2 or larger.

It is easy to see how during the iterative decoding process, after every iteration at least one symbol leaves the ripple (assuming decoding is successful). Moreover, at each iteration some output symbols might leave the cloud and enter the ripple if their reduced degree decreases from two to one. Note also that iterative decoding fails if the ripple becomes empty before k iterations. Thus, if one is able to track the size of ripple it is possible to derive the performance of LT codes under iterative decoding. In [52] a finite length analysis of LT codes is proposed that models the iterative decoder as a finite state machine, based on a dynamic programming approach. The full proof of the analysis in [52], that was published only in abstract form, can be found in [55]. This analysis can be used to derive the error probability of the iterative decoder and it also allows to compute the first order moments of the ripple and the cloud. This analysis was extended in [54], where the second moment of the ripple size was analyzed. In [53] another analysis of LT codes under iterative decoding is proposed that has lower complexity and is based on the assumption that the number of output symbol collected by the receiver follows a Poisson distribution.

3.2.1.1 Degree Distributions

In this section we present the two best well known degree distributions, the ideal soliton distribution and the robust soliton distribution. Both distributions were designed for iterative decoding.

Ideal Soliton Distribution

The first distribution we will present is known as ideal soliton distribution [25] and is based on these two design principles:

- The expected number of output symbols in the ripple at the start of iterative decoding is one.
- The expected number of output symbols leaving the cloud and entering the ripple is one at every iteration.

Thus, the expected ripple size is 1 during the whole decoding process. The ideal soliton distribution, which we denote by Ω^{ISD}, has the following expression.

$$\Omega_d^{\text{ISD}} = \begin{cases} \frac{1}{k} & d = 1 \\ \frac{1}{d(d-1)} & 1 < d \leq k. \end{cases}$$

Note that the distribution varies with the number of input symbols k. The average output degree of the ideal soliton distribution is [25]

$$\bar{\Omega}^{\text{ISD}} = H(k)$$

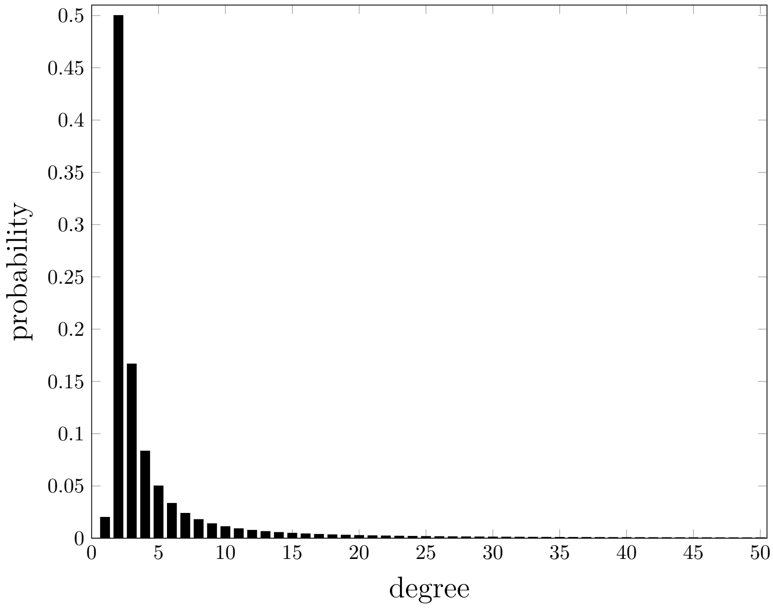


Fig. 3.8 Ideal soliton distribution, Ω^{ISD}, for $k = 50$.

where $H(k)$ is the harmonic sum up to k:

$$H(k) = \sum_{i=1}^{k} \frac{1}{i}.$$

Since, the harmonic sum can be approximated as $H(k) \approx \log(k)$, we can approximate the average output degree of Ω^{ISD} as

$$\bar{\Omega}^{\text{ISD}} \approx \log(k),$$

For illustration we provide a plot of the ideal soliton distribution for $k = 50$ in Figure 3.8.

In practice the ideal soliton distribution does not show a good performance. The reason behind this poor performance is that its design only takes into account the *expected* value of symbols entering the ripple. In practice, however, there are statistical variations in the iterative decoding process that make the ideal soliton distribution fail with high probability.

Let us denote the probability of decoding failure by $\mathsf{P_F}$. A lower bound to $\mathsf{P_F}$ is the probability that the decoding cannot start at all because the ripple is empty (no degree one output symbols), we shall denote this probability by P_{ns}. This probability corresponds to

$$P_{\mathrm{ns}} = \binom{m}{0} \Omega_1^{\mathrm{ISD}^0} \left(1 - \Omega_1^{\mathrm{ISD}}\right)^m = \left(1 - \frac{1}{k}\right)^{k(1+\epsilon)}.$$

If we now let k (and m) tend to infinity keeping the relative receiver overhead ϵ constant, this expression simplifies to:

$$\lim_{k \to \infty} P_{\mathrm{ns}} = e^{-(1+\epsilon)}.$$

This implies the probability of decoding failure is in practice very high, since one usually wants to operate at low ϵ (the overhead should ideally be small).

Robust Soliton Distribution

The robust soliton distribution was introduced in the original LT paper from Luby, [25]. This distribution is an improvement of the ideal soliton. In fact, the design goal of the robust soliton distribution is ensuring that the expected ripple size is large enough at each point of the decoding with high probability. This ensures that iterative decoding does not get stuck in the middle of the decoding process.

The robust soliton distribution is actually a family of parametric distributions that depend on two parameters ψ and ς. Let $R = \varsigma \log(k/\psi)\sqrt{k}$. The robust soliton distribution is obtained as:

$$\Omega_d^{\mathrm{RSD}} = \frac{\Omega_d^{\mathrm{ISD}} + \tau_d}{\beta}, \tag{3.3}$$

where τ_d and β are given by

$$\tau_d = \begin{cases} \frac{R}{d\ k} & 1 \leq d \leq \frac{k}{R-1} \\ R\ \log\left(R/\psi\right)/k & d = \frac{k}{R-1} \\ 0 & d > \frac{k}{R-1} \end{cases},$$

and

$$\beta = \sum_{d=i}^{k} \Omega_d^{\mathrm{ISD}} + \tau_d.$$

Therefore, the robust soliton distribution is obtained as a mixture of the ideal soliton distribution with a correction term τ . The average output degree for this distribution can be upper bounded by [25] :

$$\bar{\Omega}^{\mathrm{RSD}} \leq H(k) + 1 + \log(R/\psi).$$

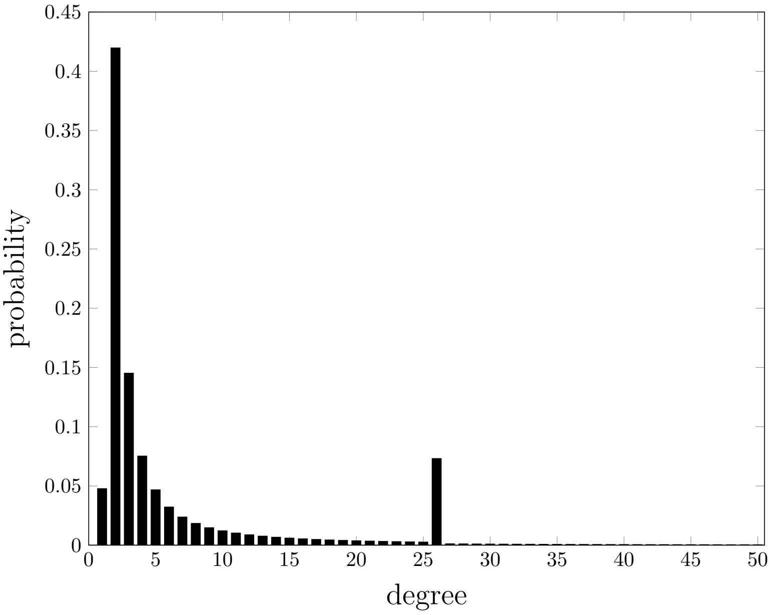


Fig. 3.9 Robust soliton distribution, Ω^{RSD}, for $k = 50$, $\psi = 0.2$ and $\varsigma = 0.05$.

For illustration we provide a plot of a robust soliton distribution in Figure 3.9. We can observe how the probability of degree one output symbols is increased with respect to the ideal soliton distribution. Moreover, a *spike* appears in the distribution at $d = k/(R-1)$.

In Figure 3.10 we provide a performance comparison for the ideal and robust soliton distribution for $k = 100$. More concretely we show the probability of decoding failure under iterative decoding, $\mathsf{P_F}$, vs. the relative receiver overhead ϵ. It can be observed how the asymptotic lower bound to $\mathsf{P_F}$ for the ideal soliton distribution holds and is actually tight for high ϵ. Moreover, we can observe how the probability of decoding failure of the robust soliton distribution is much lower than that of the ideal soliton distribution.

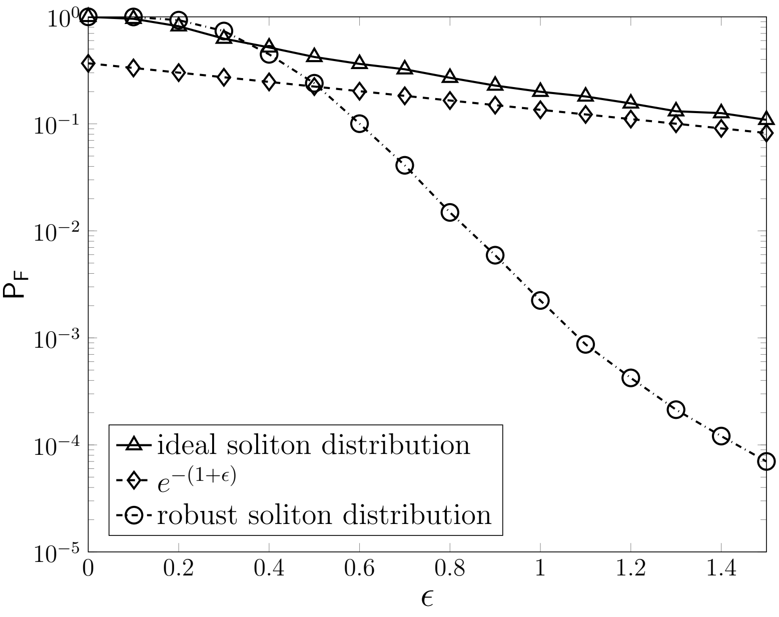


Fig. 3.10 Probability of decoding failure $\mathsf{P_F}$ vs. relative receiver overhead ϵ for the ideal and robust soliton distribution with $\psi = 0.33$ and $\varsigma = 0.234$ for $k = 100$.

3.2.2 Maximum Likelihood Decoding

As we saw in Section 3.2, for fixed n the relation between source symbols $\mathbf{v}$ and output symbols $\mathbf{c}$ can be expressed by a system of linear equations:

$$\mathbf{c} = \mathbf{v}\mathbf{G}$$

where we recall, that $\mathbf{G}$ was the generator matrix of the fixed-rate LT code. That is, under the assumption that the number of output symbols n is fixed.

Let us assume that out of the n output symbols generated by the LT encoder the receiver collects $m = k + \delta$, that we denote by $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. Denoting by $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ the set of indices corresponding to the m non-erased symbols, we have

$$y_j = c_{i_j}.$$

The dependence of the received output symbols on the source symbols can be expressed as:

$$\mathbf{y} = \mathbf{v}\tilde{\mathbf{G}} \tag{3.4}$$

with $\tilde{\mathbf{G}}$ given by the m columns of $\mathbf{G}$ with indices in $\mathcal{I}$.

LT decoding consists in finding the solution to the system of linear equations in (3.4). The solution will be unique only if $\tilde{\mathbf{G}}$ has full rank, that is, if its rank is k. If $\tilde{\mathbf{G}}$ is rank deficient the system of equations does not have a unique solution and the receiver is not able to recover all source symbols[3].

Iterative decoding is a suboptimal algorithm, it is not always able to find the solution when $\tilde{\mathbf{G}}$ has full rank. For example, if $\tilde{\mathbf{G}}$ has full rank but does not have any row with Hamming weight one (degree one output symbol), iterative decoding is unable to find the solution.

A maximum likelihood (ML) decoding algorithm is an *optimal* decoding algorithm, in the sense that it always finds the solution to the system of linear equations whenever $\tilde{\mathbf{G}}$ has full rank. Therefore the performance of any ML decoding algorithm depends only on the rank properties of $\tilde{\mathbf{G}}$ and, more concretely, on the probability of $\tilde{\mathbf{G}}$ having full rank. In [56] the performance of LT codes under ML decoding was studied and a lower bound to the probability of decoding failure $\mathsf{P_F}$ was derived:

$$\underline{\mathsf{P}}_{\mathsf{F}} = \sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} \left(\sum_{d=1}^{k} \Omega_d \frac{\binom{k-i}{d}}{\binom{k}{d}} \right)^{k(1+\epsilon)} . \tag{3.5}$$

The lower bound is very tight for reception overhead slightly larger than $\epsilon = 0$.

In practice, different ML decoding algorithms can be used to solve a system of equations and they all provide the same solution, that is unique when $\tilde{\mathbf{G}}$ is full rank. However, different ML decoding algorithms have different decoding complexity, and some algorithms are more suitable than others for practical use.

3.2.3 Complexity Considerations

So far, the only performance metric we have dealt with is the probability of decoding failure. The other important metric when dealing with any coding scheme is its complexity both in encoding and decoding. Let us define complexity as the total number of operations (xor or symbol copy) needed for encoding / decoding. Since we consider binary LT codes, we only perform xor operations, which correspond to additions over $\mathbb{F}_2$. Note that decoding also requires copying the content of output symbols into input symbols. For the sake of completeness, we shall also count symbol

[3] In this thesis we focus on problems in which it is necessary to recover all source symbols, therefore, we declare a decoding failure whenever one or several source symbols cannot be recovered.

copy as one operation. Let us also define the encoding cost as the encoding complexity normalized by the number of output symbols and the decoding cost as the decoding complexity normalized by the number of input symbols.

3.2.3.1 Encoding Complexity

Let us first consider encoding complexity. Generating an output symbol of degree d requires d operations. Thus, given a degree distribution Ω, the encoding cost will be given by the average output degree $\bar{\Omega}$. In proposition 1 we have shown how a necessary condition for decoding to be successful with high probability is $\bar{\Omega}(k) = \mathcal{O}(\log(k))$. This implies that the encoding cost will need to be at least $\mathcal{O}(\log(k))$.

3.2.3.2 Iterative Decoding Complexity

We consider now the complexity of LT iterative decoding. Let us assume a generic degree distribution Ω, with average output degree $\bar{\Omega}$ that requires a relative receiver overhead ϵ^* for decoding to be successful with high probability. If we think of a bipartite representation of our LT code, we can think of encoding as drawing the edges in the graph, where every edge implies performing one operation (xor or symbol copy). Similarly, iterative decoding starts operating on a bipartite graph containing the $m = (1 + \epsilon^*)k$ received output symbols and k input symbols. During iterative decoding edges are erased from the graph, being each edge again associated to one operation. At the end of iterative decoding all edges are erased from the graph[4]. Thus the decoding cost under iterative decoding corresponds to $(1 + \epsilon^*)\bar{\Omega}$.

In proposition 1 we have shown how a necessary condition for decoding to be successful with high probability is $\bar{\Omega}(k) = \mathcal{O}(\log(k))$. This implies that the iterative decoding cost will need to be at least $\mathcal{O}(\log(k))$.

3.2.3.3 Maximum Likelihood Decoding Complexity

Many different ML decoding algorithm exists that can be used to solve a linear system of equations. All ML algorithms lead to the same solution, that in our case is unique when matrix $\tilde{\mathbf{G}}$ is full rank. The ML decoding complexity will vary depending on which ML decoding algorithm is used.

[4] Actually, at the last iteration of iterative decoding some edges might still be present in the graph since we might have more than one output symbol in the ripple. We neglect this effect for the sake of simplicity.

The best known algorithm is probably Gaussian elimination. This algorithm has a decoding complexity of $\mathcal{O}\left(k^3\right)$ and is generally not practical for values of k beyond the hundreds. The problem of solving systems of linear equations is a well known problem that appears not only in erasure correction. Several algorithms exist that have a lower (asymptotic) complexity than Gaussian elimination. For example, the Wiedemann algorithm [57] can be used to solve sparse systems of linear equations with a complexity of $\mathcal{O}\left(k^2 \log^2(k)\right)$. In [58] different algorithms are studied to solve large systems of sparse linear equations over finite fields. In this work, the running times of different decoding algorithms are compared for systems of equations arising from integer factorization and the computation of discrete logarithms. The main finding of the paper is that if the system of equations is sparse, there exists a class of algorithms that in practice requires shorter running times than the Wiedemann algorithm when k is below 10^5. This class of algorithms is usually known as *structured* or *intelligent* Gaussian elimination. They consist of reducing the system of equations to a much smaller one than can be solved using other methods (Gaussian elimination, for example). Let us assume that Gaussian elimination is used to solve the reduced system of equations, and let us also assume that our intelligent Gaussian elimination algorithm is able to reduce the size of the system of equations from k to k/f, where $f > 1$. Since the complexity of Gaussian elimination is $\mathcal{O}\left(k^3\right)$, for large enough k, the intelligent Gaussian elimination algorithm will reduce complexity at least by a factor f^3. Despite having a higher asymptotic complexity (the complexity is still $\mathcal{O}\left(k^3\right)$) these algorithms have shorter running times than other algorithms, such as the Wiedemann algorithm (provided that f is large enough and k not too large).

The ML decoding algorithm used in practice for fountain codes is usually referred to as inactivation decoding. This algorithm belongs to the family of *structured* or *intelligent* Gaussian elimination algorithms and it will be explained in detail in Chapter 4.1.

3.2.4 Systematic LT Codes

In practical applications it is desirable that fountain codes are systematic, that is, the first k output symbols should correspond to the k input symbols. Thus, if the quality of the transmission channel is good and no erasures occur, the receiver does not need to carry out decoding. A straightforward way of making a fountain code systematic is simply transmitting the first k input symbols and afterwards start transmitting output symbols from the fountain code. We will refer to this construction as trivially

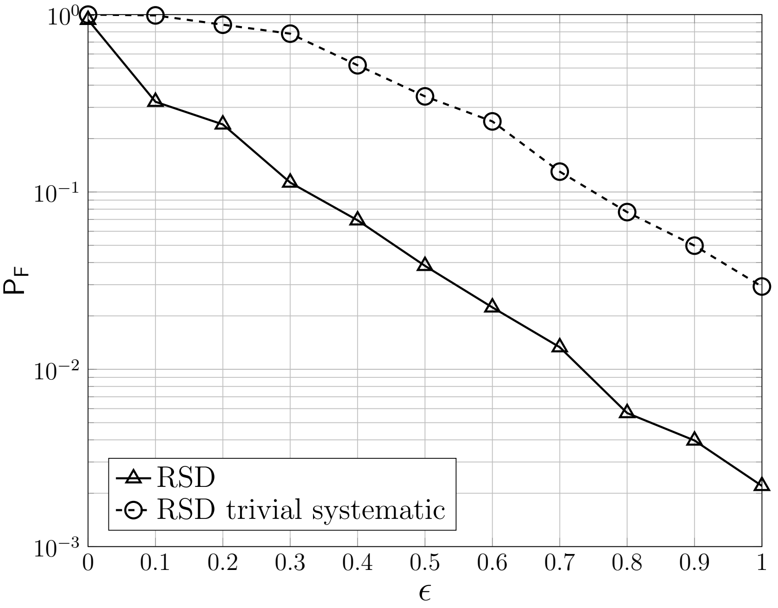


Fig. 3.11 $\mathsf{P_F}$ vs. ϵ for a robust soliton distribution with $\psi = 0.33$ and $\varsigma = 0.234$ for $k = 100$ under ML decoding. The solid line with triangle markers represents the probability of failure of a standard LT code. The dashed line with round markers represents the probability of failure of a trivially systematic LT code over a BEC with erasure probability $\varepsilon = 0.1$.

systematic LT code. This construction shows a poor performance since the receiver overhead needed to decode successfully increases substantially [59].

Figure 3.11 shows the probability of decoding failure for a robust soliton distribution (RSD) for $k = 100$ with parameters with $\psi = 0.33$ and $\varsigma = 0.234$ under ML decoding. In particular, two codes are considered, a standard LT code and a trivially systematic LT code over a BEC with erasure probability $\varepsilon = 0.1$. It can be observed the trivial systematic code performs much worse than the standard non systematic LT code.

The bad performance of trivially systematic LT codes might seem surprising at first. The intuition behind this bad performance is the following. Assume that a substantial fraction of systematic symbols are received, for example, let us assume the decoder has received μ of the systematic symbols and that the remaining fraction $1 - \mu$ have been erased. In order to be able to decode, the receiver will need to receive output symbols with neighbors within the yet unrecovered input symbols. Moreover, any output symbol having neighbors only within the received systematic symbols will be useless for decoding. Let us now assume that an output symbol of degree d is received.

The probability that all its neighbors are within the received systematic symbols is

$$\frac{\binom{\mu k}{d}}{\binom{k}{d}}.$$

Under the assumption that k is large, $k \gg d$, and that output symbols choose their neighbours with replacement, a simplified expression for this probability can be obtained. Under these assumptions, we have that the probability that one of the neighbors of an output symbol is within the received systematic symbols is μ. Hence, the probability that all d neighbors are within the received systematic symbols is μ^d. Thus, when the fraction of received systematic symbols μ is close to one, and d is not too large, most of the received output symbols will not help at all in decoding. A more detailed analysis of this effect can be found in [55].

In practice a different systematic construction is used that was patented in [59] and that will be presented next.

Let us recall that (for fixed n) LT encoding can be seen as a vector-matrix multiplication:

$$\mathbf{c} = \mathbf{v}\mathbf{G},$$

where $\mathbf{v}$ is the row vector of k input (source) symbols, $\mathbf{c}$ is the row vector of n output symbols, and $\mathbf{G}$ is an $k \times n$ binary matrix which defines the relation between the input and the output symbols (generator matrix). To construct a systematic LT code we start with an LT code with generator matrix in the shape

$$\mathbf{G} = \left[\mathbf{G}_1 | \mathbf{G}_2\right],$$

where $\mathbf{G}_1$ is a full-rank $k \times k$ matrix that corresponds to the first k output symbols and $\mathbf{G}_2$ is a $k \times (m-k)$ matrix. First, one needs to compute the inverse matrix of $\mathbf{G}_1$, ${\mathbf{G}_1}^{-1}$. The next step is computing:

$$\mathbf{w} = \mathbf{v}{\mathbf{G}_1}^{-1}.$$

Vector $\mathbf{w}$ is then used as input to the LT encoder. Thus, the output of the LT encoder will be:

$$\mathbf{c} = \mathbf{w}\mathbf{G} = \mathbf{v}{\mathbf{G}_1}^{-1}\left[\mathbf{G}_1 | \mathbf{G}_2\right] = \mathbf{v}\left[\mathbf{I} | {\mathbf{G}_1}^{-1}\mathbf{G}_2\right],$$

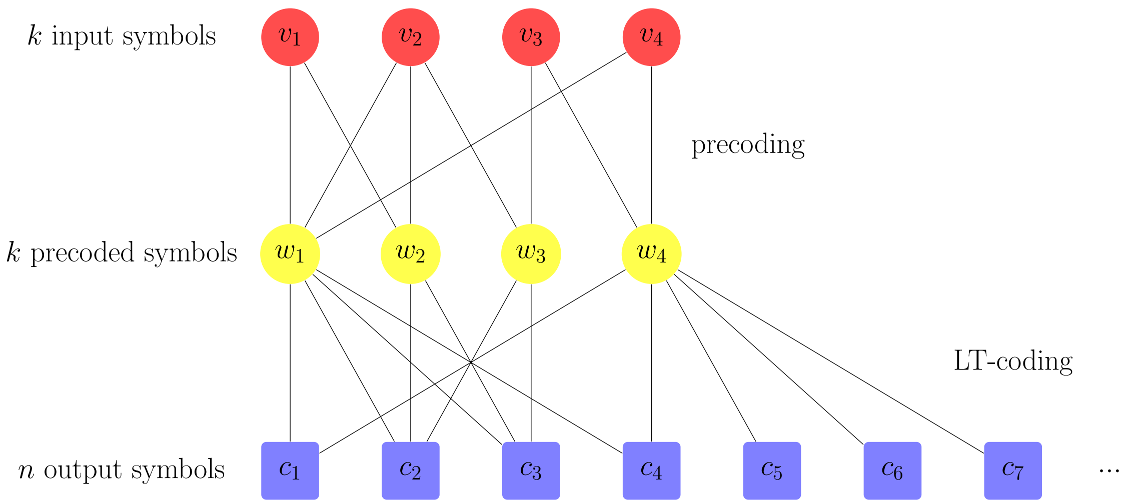


Fig. 3.12 Systematic LT code.

where $\mathbf{I}$ is the $k \times k$ identity matrix. Hence, the first k output symbols correspond to the input symbols $\mathbf{v}$. For illustration Figure 3.12 shows a graph representation of a systematic LT code.

At the decoder side two different scenarios can be considered. In case none of the first k output symbols of our systematic LT code are erased, there is obviously no need to carry out decoding. In case some erasures do occur, decoding can be done in two steps. First, standard LT decoding can be carried out to recover $\mathbf{w}$. This consists of solving the system of equations

$$\mathbf{y} = \mathbf{w}\tilde{\mathbf{G}},$$

where $\tilde{\mathbf{G}}$ is a $k \times m$ matrix that corresponds to the m columns of $\mathbf{G}$ associated to the output symbols that were not erased by the channel and $\mathbf{y}$ are the received output symbols. This system of equations can be solved in several ways, for example using iterative decoding or inactivation decoding. Finally, the input symbols can be recovered computing

$$\mathbf{v} = \mathbf{w}\mathbf{G}_1.$$

Note that this last step corresponds to LT encoding (since by construction $\mathbf{G}_1$ is sparse, this last step is actually less complex than a standard vector matrix multiplication).

The main advantage of this construction is that its performance in terms of probability of decoding failure is similar to that of non-systematic LT codes [60]. However, this comes at some cost in decoding complexity, since an additional LT encoding needs to be carried out at the decoder.

3.3 Raptor Codes

Raptor codes were originally patented in [26] and published in [27, 28]. They were also independently proposed in [29], where they are referred to as online codes. Raptor codes are an evolution of LT codes. More concretely, Raptor codes are a serial concatenation of an outer (fixed-rate) block code $\mathcal{C}$ (usually called precode) with an inner LT code.

At the input we have a vector of k input (or source) symbols, $\mathbf{u} = (u_1,\ u_2,\ \ldots, u_k)$. Out of the input symbols, the outer code generates a vector of h intermediate symbols $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_h)$, where $h > k$. Denoting by $\mathbf{G}_{\mathrm{o}}$ the employed generator matrix of the outer code, of dimension $(k \times h)$, the intermediate symbols can be expressed as

$$\mathbf{v} = \mathbf{u}\mathbf{G}_{\mathrm{o}}.$$

By definition, $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_h) \in \mathcal{C}$, i.e., the intermediate word is a codeword of the outer code $\mathcal{C}$.

The intermediate symbols serve as input to an LT code that can generate an unlimited number of output symbols, $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, where n can grow unbounded. Hence, Raptor codes inherit the rateless property of LT codes. For any n the output symbols can be expressed as

$$\mathbf{c} = \mathbf{v}\mathbf{G}_{\mathrm{LT}} = \mathbf{u}\mathbf{G}_{\mathrm{o}}\mathbf{G}_{\mathrm{LT}}$$

where $\mathbf{G}_{\mathrm{LT}}$ is the generator matrix of the (fixed-rate) LT code. Hence, $\mathbf{G}_{\mathrm{LT}}$ is an $(h \times n)$ binary matrix, each column of $\mathbf{G}_{\mathrm{LT}}$ being associated with a received output symbol as seen in Section 3.2.

Figure 3.13 shows a graph representation of a Raptor code, where the input symbols are represented as green diamond-shaped nodes, the intermediate symbols as red circular nodes and the output symbols as blue squared nodes.

The design principle of Raptor codes can be intuitively explained as follows. In Chapter 3.2 we saw that a necessary condition for LT codes to be successfully decoded with high probability is that the average output degree is $\mathcal{O}\left(\log(k)\right)$. This implies an encoding cost of $\mathcal{O}\left(\log(k)\right)$ and a decoding cost of $\mathcal{O}\left(\log(k)\right)$ as well (under iterative decoding). The main idea behind Raptor codes is relaxing the requirements on the LT

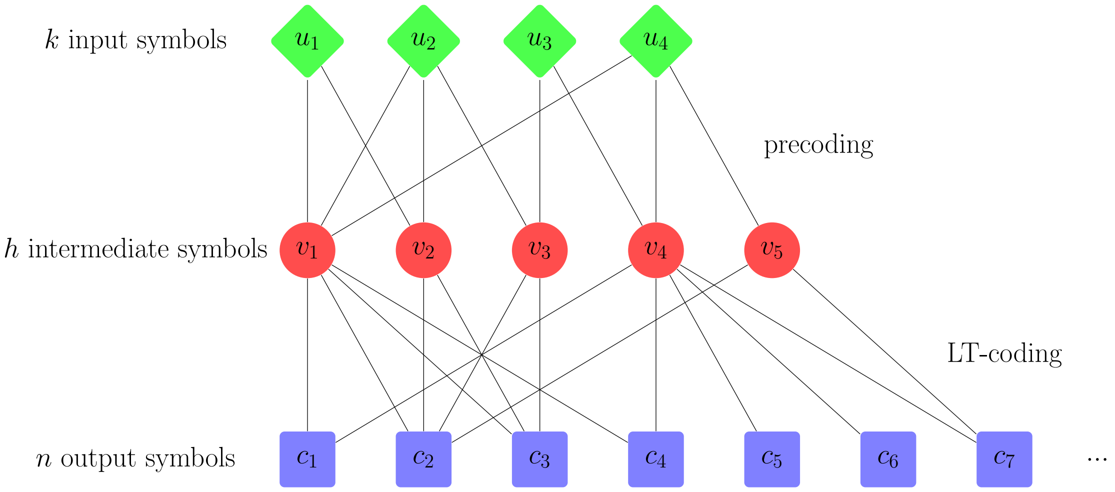


Fig. 3.13 Graph representation of a Raptor code.

code. Instead of requiring that the LT recovers *all* its input symbols, the inner LT code of a Raptor code is only required to recover with high probability a constant fraction $1 - \zeta$ of the intermediate symbols. This can be achieved with a constant average output degree. Let us assume that k is large and that the receiver has collected m output symbols. From the proof of Prop. 1, we have that for asymptotically large h the fraction of intermediate symbols with no edges attached (uncovered) will correspond to:

$$\zeta = e^{-\bar{\Omega}\frac{m}{h}}.$$

Let us assume that all the covered intermediate symbols can be recovered by the LT code. The uncovered intermediate symbols can be considered as erasures by the outer code. If the outer code is an erasure correcting code that can recover with high probability from a fraction of ζ erasures, we will be able to recover all input symbols with high probability.

If the precode $\mathcal{C}$ is linear-time encodable, then the Raptor code has a linear encoding complexity, $\mathcal{O}(k)$ since the LT code has constant average output degree (i.e., the average output degree does not increase with k). Therefore, the overall encoding cost per output symbol is constant with respect to k. If the precode also accepts a linear time decoding algorithm (iterative decoding), and the LT code is decoded using iterative decoding, the decoding complexity is also linear. Hence, the decoding cost per symbol is constant. Furthermore, already in the original Raptor code paper [28], Shokrollahi showed that Raptor codes under iterative decoding are universally capacity-achieving on the binary erasure channel. Hence, they achieve the capacity of any erasure channel no matter which erasure probability the channel has.

3.3.1 Raptor Decoding

The output symbols $\mathbf{c}$ generated by the Raptor encoder are transmitted over a BEC at the output of which each transmitted symbol is either correctly received or erased. Let us denote by m the number of output symbols collected by the receiver of interest, where $m = k + \delta$, being δ the absolute receiver overhead. We denote by $\mathbf{y} = (y_1, y_2, \dots, y_m)$ the m received output symbols. Denoting by $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ the set of indices corresponding to the m non-erased symbols, we have

$$y_j = c_{i_j}.$$

The relation between the received output symbols and the input symbols can be expressed as:

$$\mathbf{y} = \mathbf{v}\mathbf{G}_{\mathrm{R}} \tag{3.6}$$

where

$$\mathbf{G}_{\mathrm{R}} = \mathbf{G}_{\mathrm{o}}\tilde{\mathbf{G}}_{LT}$$

with $\tilde{\mathbf{G}}_{LT}$ given by the m columns of $\mathbf{G}_{\mathrm{LT}}$ with indices in $\mathcal{I}$.

Raptor decoding consist of recovering the input symbols $\mathbf{v}$ given the received output symbols $\mathbf{y}$. Although it is possible to perform Raptor decoding by solving the linear system of equations in (3.6), this is not done in practice for complexity reasons. The decoding algorithms employed in practice, iterative decoding or inactivation decoding, require that the system of equations is sparse in order to show good performance and matrix $\mathbf{G}_{\mathrm{R}}$ is not sparse in general.

In practice, instead of the generator matrix of the Raptor code, another matrix representation is used that is usually referred to as constraint matrix, since it is an alternative representation of the coding constraints of the outer and inner code. The constraint matrix of a Raptor code is defined as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{H}_{\mathrm{o}} \\ \tilde{\mathbf{G}}_{LT}^{T} \end{bmatrix},$$

where $\mathbf{H}_\text{o}$ is the parity check matrix of the outer code (precode) with size $((h-k)\times h)$. Thus, $\mathbf{M}$ is a $((h-k+m)\times h)$ binary matrix.

By definition, the intermediate word of a Raptor code is a codeword of the precode, $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_h) \in \mathcal{C}$. Hence, one can write

$$\mathbf{H}_\text{o}\ \mathbf{v}^T = \mathbf{z} \tag{3.7}$$

where $\mathbf{z}$ is a zero column vector of size $((h-k)\times 1)$. Similarly, one can express the vector of received output symbols $\mathbf{y}$ as:

$$\tilde{\mathbf{G}}_{LT}^T\ \mathbf{v}^T = \mathbf{y}^T. \tag{3.8}$$

Putting together (3.7) and (3.8), we have

$$\mathbf{M}\ \mathbf{v}^T = \begin{bmatrix} \mathbf{z} \\ \mathbf{y}^T \end{bmatrix}. \tag{3.9}$$

In practical Raptor decoders (3.9) is used for decoding. The main advantage of the constraint matrix is that it preserves the sparsity of the generator matrix of the LT code. Moreover, it also preserves the sparsity of the parity check matrix of the precode, in case it is sparse.

The system of equations in (3.9) can be solved using different techniques, such as iterative decoding, standard Gaussian elimination or inactivation decoding. Similarly to LT codes, most works on Raptor codes consider large input blocks (k at least in the order of a few tens of thousands symbols) and iterative decoding. However, in practice smaller blocks are used, usually due to memory limitations at the decoders. For example, in the most widespread binary Raptor codes, R10 (release 10), values of k ranging from 1024 to 8192 are recommended (see Section 3.3.2). For these input block lengths, the performance of iterative decoding suffers a considerable degradation. Therefore, instead of iterative decoding, ML decoding is used (inactivation decoding).

3.3.2 R10 Raptor Codes

The state of the art binary Raptor code is the R10 (release 10) Raptor code. This code is systematic and was designed to support a number of input symbols ranging from $k = 4$ to $k = 8192$ [60]. The maximum supported number of output symbols is

$n = 65,536$. The probability of decoding failure $\mathsf{P_F}$ shows an error floor lower than 10^{-6} for all values of k [60].

The precode used by R10 Raptor codes is a serial concatenation of two systematic erasure correcting codes.

- The outer code is a systematic low-density parity-check (LDPC) code that introduces s_{R10} redundant symbols. The number of LDPC redundant symbols is a function of k and can be approximated as [60]

  $$s_{\mathrm{R10}} \approx 0.01k + \sqrt{2k}.$$

  Its parity check matrix is composed of $\lceil k/s_{\mathrm{R10}} \rceil$ degree 3 circulant matrices plus a $(s_{\mathrm{R10}} \times s_{\mathrm{R10}})$ identity matrix. The Hamming weight of each of the rows of the parity check matrix of the LDPC code is approximately $3\lfloor k/s_{\mathrm{R10}} \rceil + 1$, where $\lfloor x \rceil$ denotes the closest integer to x (the last circulant matrix might not be complete).

- The inner code is a systematic high-density parity-check (HDPC) code that introduces h_{R10} redundant symbols. The number of HDPC redundant symbols depends on k approximately as [60]

  $$h_{\mathrm{R10}} \approx \log_2(1.01k + \sqrt{2k}).$$

  Its parity check matrix is composed of a dense part and a $(h_{\mathrm{R10}} \times h_{\mathrm{R10}})$ identity matrix. The dense part is obtained from a binary reflected Gray code and has the property that the normalized Hamming weight of every row is approximately $1/2$. Therefore, roughly half of the elements in the dense part of the parity check matrix are set to 1.

Thus, the total number of intermediate symbols h corresponds to

$$h = k + s_{\mathrm{R10}} + h_{\mathrm{R10}}.$$

Figure. 3.14 shows the number of redundant LDPC and HDPC symbols, s_{R10} and h_{R10}, as a function of k. It can be observed how for all values of k except for very small values ($k = 4$) the number of LDPC redundant symbols is higher than that of HDPC redundant symbols. Therefore, the rate of the LDPC code is lower than that of the HDPC code, as it can be observed in Figure. 3.15. In this last figure it can

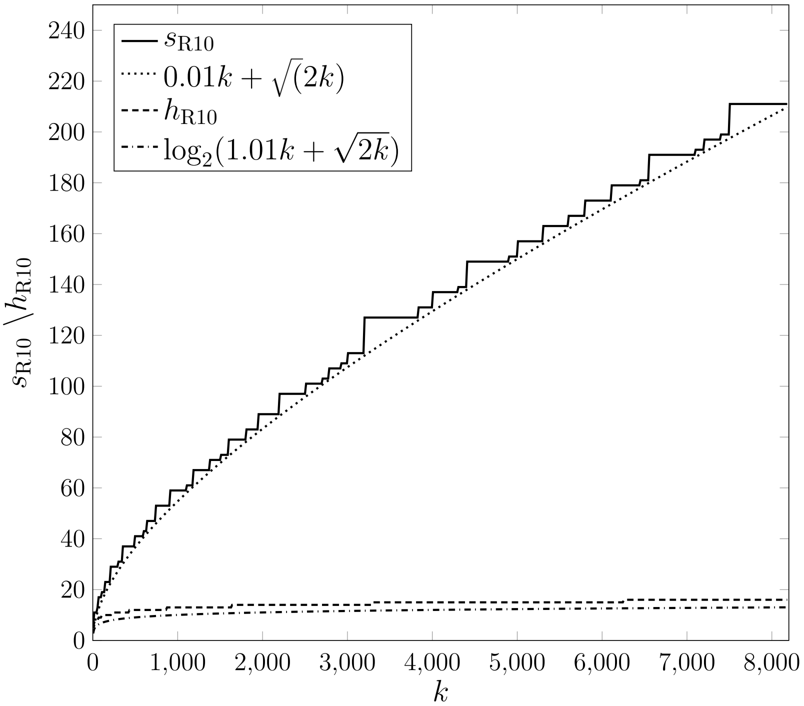


Fig. 3.14 Number of LDPC (s_{R10}) and HDPC (h_{R10}) redundant symbols in R10 Raptor codes vs. k and their approximate values.

also be observed how the outer code rate increases with k, although it does not do it monotonically.

The precode of the R10 Raptor code was designed to behave similarly to a uniform random matrix in terms of rank properties but admitting a fast matrix vector multiplication algorithm [60].

The degree distribution of the LT code is given by:

$$\begin{aligned}\Omega^{\mathrm{R10}}(\mathrm{x}) = 0.0098\mathrm{x} + 0.4590\mathrm{x}^2 + 0.2110\mathrm{x}^3 + 0.1134\mathrm{x}^4 \\ + 0.1113\mathrm{x}^{10} + 0.0799\mathrm{x}^{11} + 0.0156\mathrm{x}^{40}.\end{aligned} \tag{3.10}$$

Its average output degree is $\bar{\Omega} = 4.631$.

For illustration in Figure. 3.16 we provide the constraint matrix for a R10 Raptor code for $k = 20$ and $m = 30$. In this case $s_{\mathrm{R10}} = 11$ and $h_{\mathrm{R10}} = 7$. In the upper part, highlighted in blue, the parity check matrix of the LDPC code can be distinguished. This submatrix is composed of two circulant matrices and an identity matrix. All rows of this submatrix have Hamming weight 6 or 7. Below it the parity check matrix of

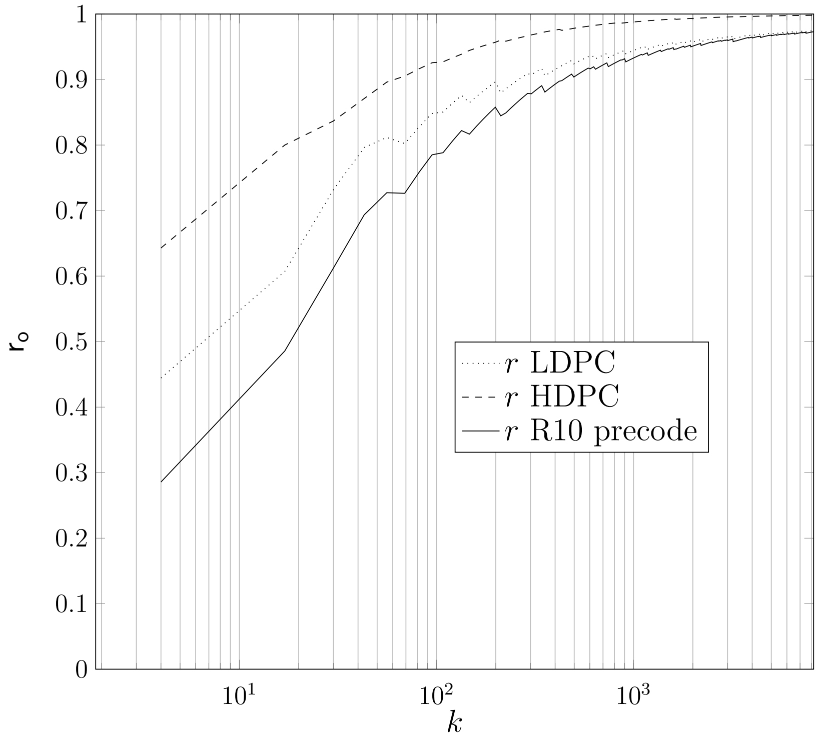


Fig. 3.15 Precode rate $\mathsf{r_o}$ vs. k for R10 Raptor codes. The rate of the LDPC and HDPC codes is also shown.

the HDPC code can be observed highlighted in green. These rows have a normalized Hamming weight of around 1/2 and are the densest in the constraint matrix. The lower part of the constraint matrix, highlighted in red, corresponds to the LT symbols and is sparse.

R10 Raptor codes are the state-of-the-art binary Raptor codes and are widely used in practice. In fact, they are part of several standards [60]:

- 3GPP Multimedia Broadcast Service, [30]. Raptor codes are used in a terrestrial cellular network for file delivery and streaming applications.
- Internet Engineering Task Force (IETF) RFC 5053, [61]. The R10 Raptor code is used for file delivery over data networks (i.e. the Internet). For example, this standard can be used to deliver data to a user via unicasting, or several user via multicasting, using, for instance, the User Datagram Protocol (UDP).
- Digital Video Broadcasting. The R10 Raptor code is used in several DVB standards:

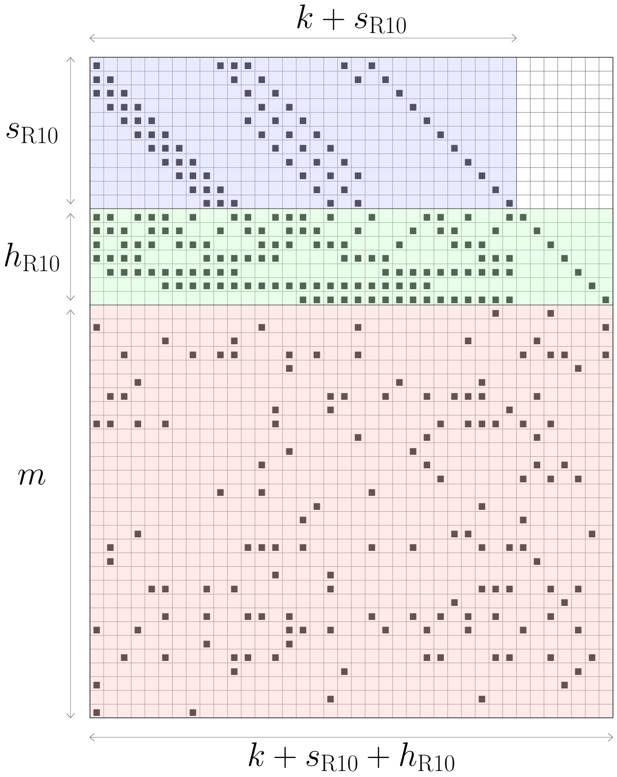


Fig. 3.16 Constraint matrix of a R10 Raptor code for $k = 20$ and $m = 30$. The blue and red sub-matrices represents respectively the parity check matrices of the LDPC and HDPC codes. The red sub-matrix represents the transposed generator matrix of the LT code. The entries filled with a square in the matrix represent the matrix elements set to one, and the empty entries those elements set to zero.

– In DVB-SH the R10 Raptor code is used as upper layer forward error correction (FEC) in order to overcome long deep fading events, [62]. Concretely, R10 Raptor codes can be used to protect Multi Protocol Encapsulation (MPE) fragments at the link layer, or User Datgram Protocol (UDP) datagrams at the transport layer. This is specially appealing for mobile satellite systems in which terminals suffer frequently of very deep fades in the received signal due to blockage by a building or tunnel, for example.

– In DVB-H, [63], R10 Raptor codes are used for similar purposes as in DVB-SH.

– DVB has also standardized the R10 Raptor codes for streaming services over IP networks, [64].

  - The R10 Raptor code is also part of a standard for broadcast/multicast data delivery, [65].

- International Telecommunication Union (ITU) IPTV services [66]. The R10 Raptor code is used for streaming applications.

R10 Raptor codes are not truly rateless, since the number of output symbols is limited to $n = 65,536$. However, for most practical scenarios they can be considered to be rateless. In spite of their (almost) rateless capability, R10 Raptor codes represent an excellent solution also for fixed-rate communication schemes requiring powerful erasure correction capabilities with low decoding complexity. In fact, in some of cases they are actually used in a fixed-rate setting (see, e.g., [62]).

3.3.3 Systematic Raptor Codes

Systematic Raptor codes are obtained similarly to systematic LT codes [59]. If we recall, Raptor code output symbols are obtained in two stages. First a vector of intermediate symbols $\mathbf{v}$ is obtained from the vector of input symbols $\mathbf{u}$ using an outer block code (precode):

$$\mathbf{v} = \mathbf{u}\mathbf{G}_\text{o},$$

where $\mathbf{G}_\text{o}$ is $(k \times h)$ generator matrix of the precode. The output symbols $\mathbf{c}$ are then obtained through an LT encoding of the intermediate symbols:

$$\mathbf{c} = \mathbf{v}\mathbf{G}_\text{LT},$$

where $\mathbf{G}_\text{LT}$ is the generator matrix of the LT code. Thus, the relationship between input and output symbols can be expressed as

$$\mathbf{c} = \mathbf{u}\mathbf{G}_\text{o}\mathbf{G}_\text{LT}.$$

A systematic Raptor code can be obtained starting with an LT generator matrix in the shape:

$$\mathbf{G}_\text{LT} = \left[\mathbf{G}_\text{LT,1} | \mathbf{G}_\text{LT,2}\right],$$

where $\mathbf{G}_\text{LT,1}$ has dimension $h \times k$ and $\mathbf{G}_\text{LT,2}$ has dimension $h \times (n-k)$. Furthermore, matrix $\mathbf{G}_\text{LT,1}$ must be chosen so that matrix

$$\mathbf{F} = \mathbf{G}_\text{o}\mathbf{G}_\text{LT,1}$$

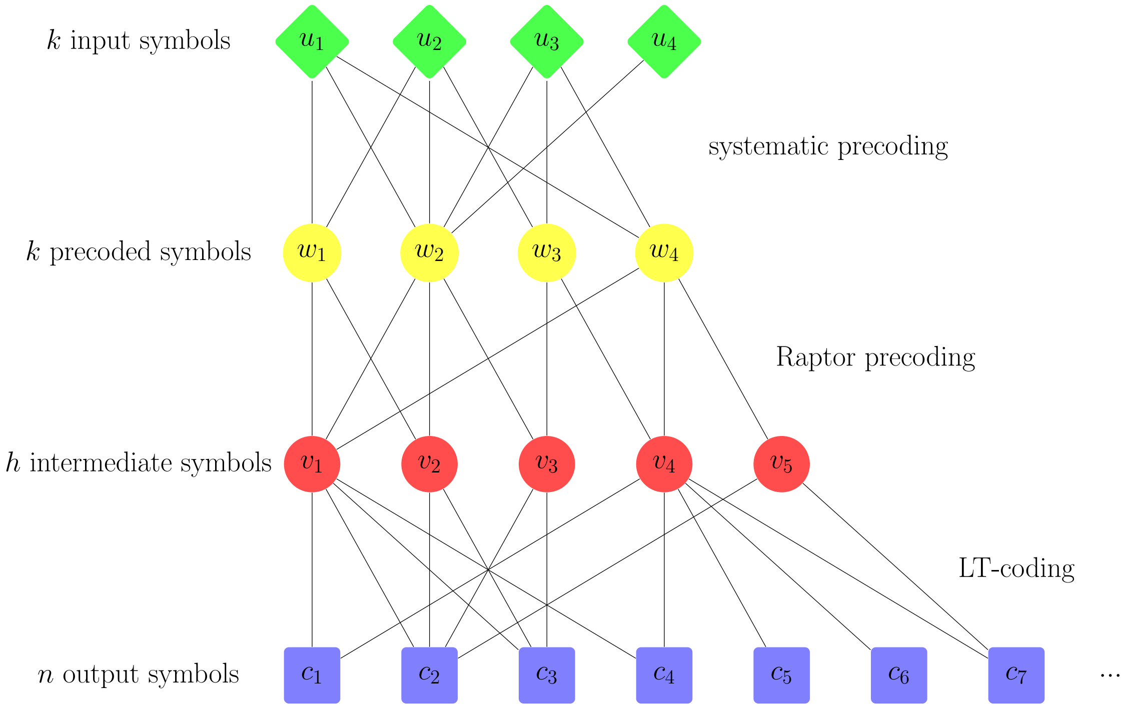


Fig. 3.17 Systematic Raptor code.

is full rank (has rank k). A systematic Raptor code is obtained by computing

$$\mathbf{w} = \mathbf{v}\mathbf{F}^{-1},$$

and using vector $\mathbf{w}$ as input to the Raptor code. Hence, the intermediate symbols of the systematic Raptor code will correspond to:

$$\mathbf{v} = \mathbf{w}\mathbf{G}_{\mathrm{o}} = \mathbf{v}\mathbf{F}^{-1}\mathbf{G}_{\mathrm{o}},$$

and its output will correspond to

$$\mathbf{c} = \mathbf{v}\mathbf{F}^{-1}\mathbf{G}_{\mathrm{o}}\mathbf{G}_{\mathrm{LT}} = \mathbf{v}\mathbf{F}^{-1}\mathbf{G}_{\mathrm{o}}\left[\mathbf{G}_{\mathrm{LT},1}|\mathbf{G}_{\mathrm{LT},2}\right] = \mathbf{v}\left[\mathbf{I}|\mathbf{F}^{-1}\mathbf{G}_{\mathrm{o}}\mathbf{G}_{\mathrm{LT},2}\right],$$

where $\mathbf{I}$ is the identity matrix of dimension k.

For illustration we provide a graph representation of a systematic Raptor code in Figure 3.17.

When decoding a systematic Raptor code, again two cases can be differentiated. In the first none of the first k output symbols are erased and decoding does not need to be carried out. In the second case some of the first k output symbols are erased. In this case the input symbols can be recovered in two stages. First standard Raptor decoding is used to recover $\mathbf{w}$. Then, the input symbols can be recovered as:

$$\mathbf{v} = \mathbf{w}\mathbf{F} = \mathbf{w}\mathbf{G}_{\mathrm{o}}\mathbf{G}_{\mathrm{LT},1}.$$

This second stage corresponds to Raptor encoding. First the output of the precode is computed and then an LT code is applied.

This systematic construction of Raptor codes achieves a similar performance to that of non-systematic Raptor codes [60].

Chapter 4

LT Codes under Inactivation Decoding

In this chapter we consider LT codes under inactivation decoding. In Section 4.1 we explain in detail inactivation decoding. Section 4.2 focuses on analyzing inactivation decoding of LT codes. Concretely Section 4.2.1 presents an analysis based on a dynamic programming approach that provides the first moment of the number of inactivations. This analysis is then extended in Section 4.2.2 to obtain the probability distribution of the number of inactivations. In Section 4.2.3 a low complexity approximate analysis of LT codes under inactivation decoding is presented. Section 4.3 shows how the results in this chapter can be used in order to design LT codes by means of an example. Finally, Section 4.4 presents a summary of the chapter.

4.1 Inactivation Decoding

Inactivation decoding is a ML decoding algorithm that is characterized by a manageable decoding complexity and is widely used in practice [32], [30]. This algorithm belongs to the family of *structured* or *intelligent* Gaussian elimination algorithms since it aims at reducing the size of the system of equations that needs to be solved.

We will describe inactivation decoding in more detail by means of an example. As explained in Section 3.2, LT decoding consists of solving the system of equations given in (3.4), which we replicate here for the sake of completeness,

$$\mathbf{y} = \mathbf{v}\tilde{\mathbf{G}},$$

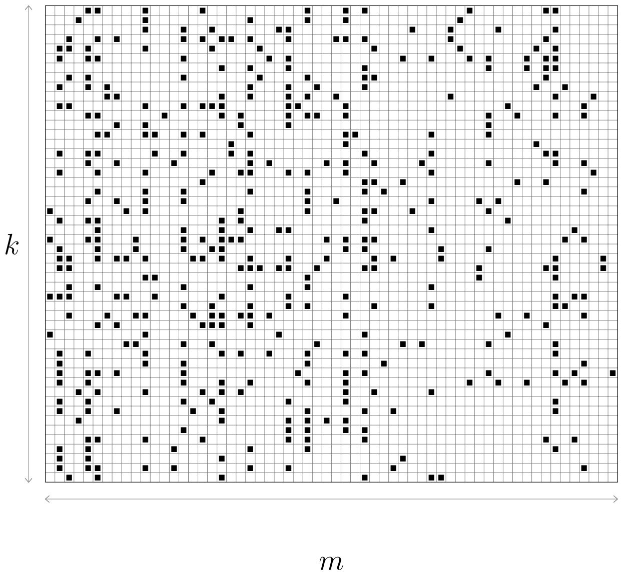


Fig. 4.1 Structure of $\tilde{\mathbf{G}}$ before inactivation decoding starts.

where we recall, $\mathbf{y}$ is the $(1 \times m)$ vector of received output symbols, $\mathbf{v}$ is the $(1 \times k)$ vector of input symbols, and $\tilde{\mathbf{G}}$ is a $k \times m$ matrix that corresponds to the m columns of $\mathbf{G}$ associated to the output symbols that were not erased by the channel. We will consider an example with $k = 50$, $m = 60$ and with $\tilde{\mathbf{G}}$ as shown in Figure 4.1. In the figure the squares inside a cell represent the elements of $\tilde{\mathbf{G}}$ that are set to 1 and the empty cells the elements that are set to 0. As it can be observed in the figure, matrix $\tilde{\mathbf{G}}$ is sparse.

Inactivation decoding consists of 4 steps:

1. *Triangulation.* $\tilde{\mathbf{G}}$ is put in an approximate upper triangular form by means of column and row permutations. Since no operation is performed on the rows or columns of $\tilde{\mathbf{G}}$, the overall density $\tilde{\mathbf{G}}$ does not change. At the end of this process we can distinguish 4 sub-matrices in $\tilde{\mathbf{G}}$. As shown in Figure 4.2 in the left upper part of $\tilde{\mathbf{G}}$ we have matrix $\mathbf{A}$ that is an upper triangular matrix of size $(k - \alpha) \times (k - \alpha)$. In the upper right part we have matrix $\mathbf{B}$ that has size $(k - \alpha) \times (m - k - \alpha)$. Finally at the lower left and right parts we have matrices $\mathbf{C}$, and $\mathbf{D}$ of respective sizes $\alpha \times (m - k - \alpha)$ and $\alpha \times (k - \alpha)$. The α last (lowest) rows of $\tilde{\mathbf{G}}$ corresponding to matrices $\mathbf{C}$ and $\mathbf{D}$ are usually referred to as inactive rows.

2. *Zero matrix procedure.* The matrix $\mathbf{A}$ is put in a diagonal form and matrix $\mathbf{B}$ is zeroed out through column sums. When one performs row/column additions on

a sparse matrix, the density of the matrix tends to increase. In our case, matrices $\mathbf{C}$ and $\mathbf{D}$ become denser as shown in Figure 4.3.

3. *Gaussian elimination (GE).* GE is applied to solve the systems of equations $\tilde{\mathbf{y}} = \tilde{\mathbf{v}}\mathbf{C}'$, where the symbols in $\tilde{\mathbf{v}}$ are called *inactive variables* and are associated with the rows of the matrix $\mathbf{C}'$ in Figure 4.3 and $\tilde{\mathbf{y}}$ are known terms associated with the columns of the matrix $\mathbf{C}'$ in Figure 4.3. This step drives the cost of inactivation decoding since its complexity is $\mathcal{O}\left(\alpha^3\right)$, cubic in the number of inactive rows α. At the end of the GE step, matrix $\tilde{\mathbf{G}}$ has the structure shown in Figure 4.4.

4. *Back-substitution.* Once the values of the inactive variables have been determined, back-substitution is applied to compute the values of the remaining variables in $\mathbf{v}$. This corresponds to setting to zero all elements of matrix $\mathbf{D}'$ in Figure 4.4. After back-substitution ends all source symbols have been recovered and, therefore, $\tilde{\mathbf{G}}$ is in reduced echelon form as shown in Figure 4.5.

A unique solution to the system of equations only exists if matrix $\tilde{\mathbf{G}}$ has full rank. If we look at Figure 4.3 it is easy to see how matrix $\mathbf{A}'$ has full rank since it is an identity matrix. Hence, matrix $\tilde{\mathbf{G}}$ has full rank only if submatrix $\mathbf{C}'$ has full rank.

Among the 4 steps of inactivation decoding, the one having the highest (asymptotic) complexity is GE. However, one has to consider that the size of the system of equations that needs to be solved by means of GE is determined by the triangulation step. Therefore, if we want to analyze the complexity of inactivation decoding we need to have a closer look at triangulation.

In order to get a better understanding of triangulation we will use a bipartite graph representation of the LT code, similar to the one we used for iterative decoding. The triangulation step can be represented by an iterative pruning of the bipartite graph of the LT code. At each iteration, a reduced graph is obtained that corresponds to a sub-graph of the original LT code graph. This sub-graph involves only a subset of the input symbols (that we call *active* input symbols) and their neighbors. In this context, we use the term *reduced* degree of output symbol c to refer to the degree of output symbol c in the reduced graph, and we will denote it by $\deg_r(c)$. Therefore, the reduced degree of a node (symbol) is less or equal to its (original) degree. Note that the reduced degree has been defined in the context of iterative decoding (definition 5). The difference is that for iterative decoding the unresolved input symbols where considered, and now we consider the active input symbols.

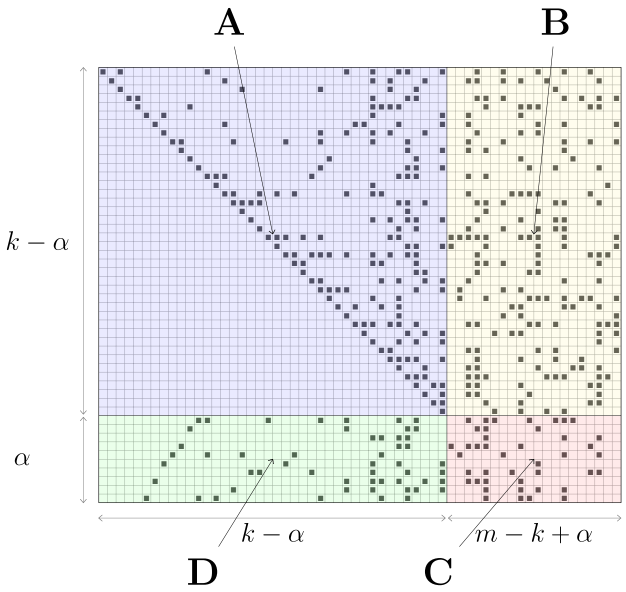


Fig. 4.2 Structure of $\tilde{\mathbf{G}}$ after the triangulation process.

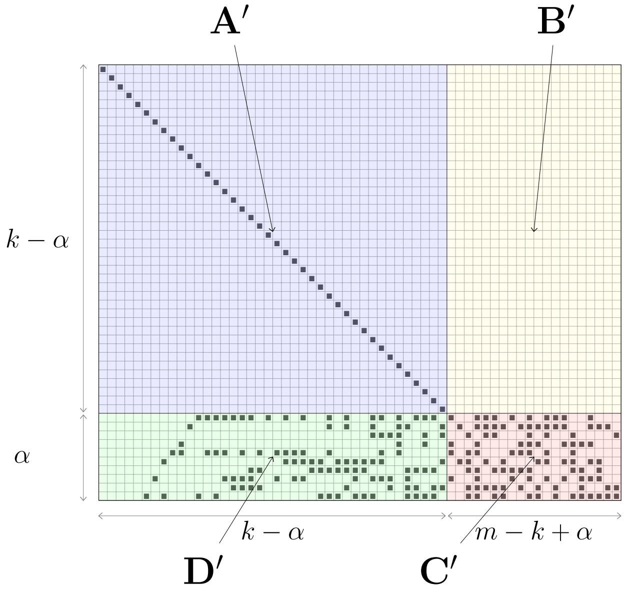


Fig. 4.3 Structure of $\tilde{\mathbf{G}}$ after the zero matrix procedure.

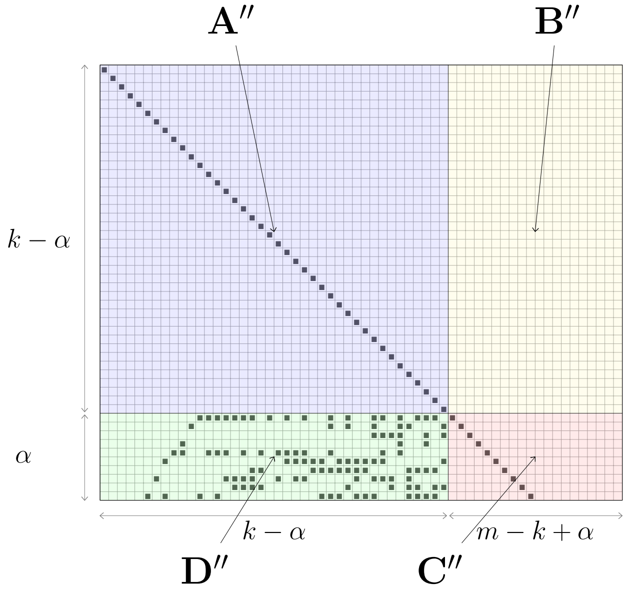


Fig. 4.4 Structure of $\tilde{\mathbf{G}}$ after Gaussian elimination.

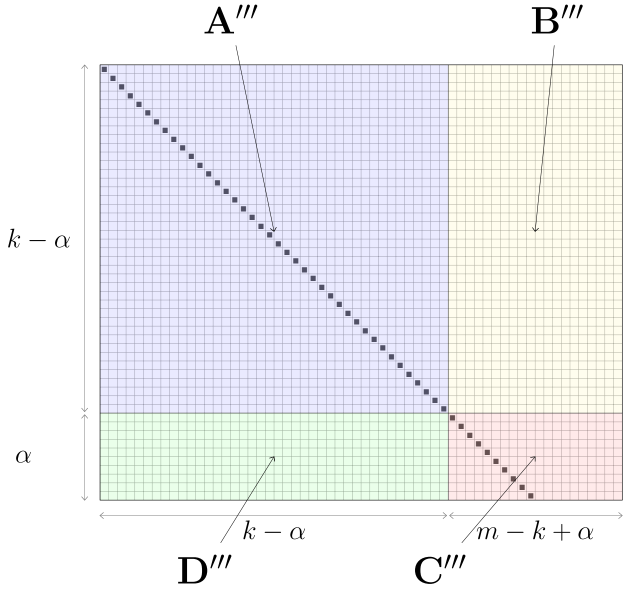


Fig. 4.5 Structure of $\tilde{\mathbf{G}}$ after back-substitution.

The triangulation process and iterative decoding are related to each other since both algorithms consist of an iterative pruning of the bipartite graph. In fact, one can think of inactivation decoding as an extension of iterative decoding. When iterative decoding is used, at every decoding step one needs to have at least one output symbol in the ripple so that decoding can go on. That is, there always needs to be at least one output symbol of reduced degree one. If the ripple becomes empty at some step, iterative decoding fails. What inactivation decoding does is restarting the iterative decoding process whenever it gets blocked (empty ripple), and does so by marking one of the active input symbols as inactive. The rational behind an inactivation is that hopefully some output symbol of reduced degree two will get its degree reduced and become of reduced degree one (it will enter the ripple), so that iterative decoding can continue. Similarly as for iterative decoding, the concepts of ripple and cloud are fundamental to understand the triangulation process. Let us recall that the ripple, $\mathscr{R}$, is defined as the set of output symbols of reduced degree 1, while the cloud, $\mathscr{W}$, corresponds to the set of output symbols of reduced degree $d \geq 2$ (see Definitions 6 and 7). Let us introduce some notation related to the cardinality of the ripple and cloud. The cardinality of the ripple will be denoted by $\mathtt{r}$ and the corresponding random variable as $\mathtt{R}$. The cardinality of the cloud will be denoted by $\mathtt{w}$ and the corresponding random variable as $\mathtt{W}$.

Triangulation starts operating on the complete bipartite graph of the LT code. Thus, before triangulation starts all source symbols are marked as active. At every step of the process, triangulation marks exactly one active source symbol as either *resolvable* or *inactive* and the symbol leaves the reduced graph. After k steps the reduced graph will correspond to an empty graph. In the following, in order to keep track of the steps of the triangulation procedure we will add a temporal dimension through the subscript u. This subscript u corresponds to the number of active input symbols in the graph. Given the fact that the number of active symbols decreases by 1 at each step, triangulation will start with $u = k$ active symbols and it will end after k steps with $u = 0$. Therefore the subscript decreases as the triangulation procedure progresses.

The following algorithm describes the triangulation procedure at step u (i.e., in the transition from u to $u - 1$ active symbols):

Algorithm 2 (Triangulation with random inactivations)**.**

- *If the ripple* $\mathscr{R}_u$ *is not empty* ($\mathtt{r}_u > 0$)

  The decoder selects an output symbol $y \in \mathscr{R}_u$ *uniformly at random. The only neighbor of* y*, i.e. the input symbol* v*, is marked as resolvable and leaves the reduced graph. The edges attached to* v *are removed.*

- *If the ripple* $\mathscr{R}_u$ *is empty* ($\mathtt{r}_u = 0$)

  An inactivation takes place. One of the active input symbols, v*, is chosen uniformly at random. This input symbol is marked as inactive and leaves the reduced graph. The edges attached to* v *are removed.*

Note that choosing the input symbol to be inactivated at random is certainly not the only possible inactivation strategy. However, this strategy makes the analysis trackable. For an overview of the different inactivation strategies we refer the reader to Section 4.1.1.

At the end of the procedure, the source symbols which are marked as resolvable correspond to the rows of matrices **A** and **B** in Figure 4.2. Similarly, the source symbols marked as inactive correspond to the rows of matrices **C** and **D**.

In order to illustrate Algorithm 2 (triangulation) we provide an example for an LT code with $k = 4$ source symbols and $m = 4$ output symbols. Before triangulation starts all source symbols are active. Figure 4.6 shows the bipartite graph of our LT code before triangulation starts. In the graph we can see how all 4 source symbols are active. If we now look at the output symbols we can see how the ripple and the cloud are composed of two elements each, $\mathscr{R} = \{y_1, y_4\}$ and $\mathscr{W} = \{y_2, y_3\}$.

Triangulation operates as follows:

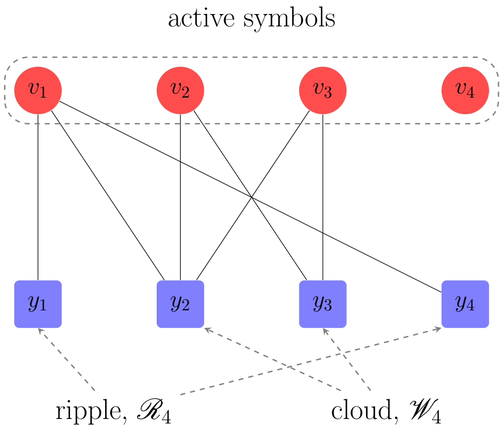


Fig. 4.6 Triangulation procedure example, $u = 4$.

1. Transition from $u = 4$ to $u = 3$. At the initial step $u = 4$, there are two output symbols in the ripple, $\mathtt{r}_4 = 2$ (see Figure 4.6). Hence, in the transition to $u = 3$ one of the source symbols (v_1) is marked as resolvable, it leaves the graph and all its attached edges are removed. The graph obtained after the transition from $u = 4$ to $u = 3$ is shown in Figure 4.7. We can see how nodes y_1 and y_4 have left the graph since their reduced degree became zero.

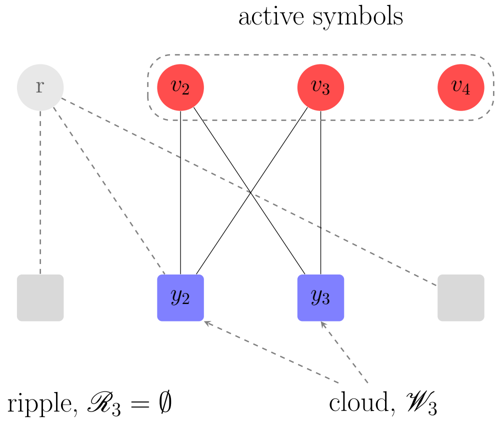


Fig. 4.7 Triangulation procedure example, $u = 3$.

2. Transition from $u = 3$ to $u = 2$. In Figure 4.7 we can see how now the ripple is empty, $\mathtt{r}_3 = 0$. Therefore, in the transition to $u = 2$ an inactivation takes place. Node v_2 is chosen at random and is marked as inactive. All edges attached to v_2

are removed from the graph. As a consequence the nodes y_2 and y_3 that were in the cloud $\mathscr{W}_3$ become of reduced degree 1 and enter the ripple $\mathscr{R}_2$. This can be observed in Figure 4.8.

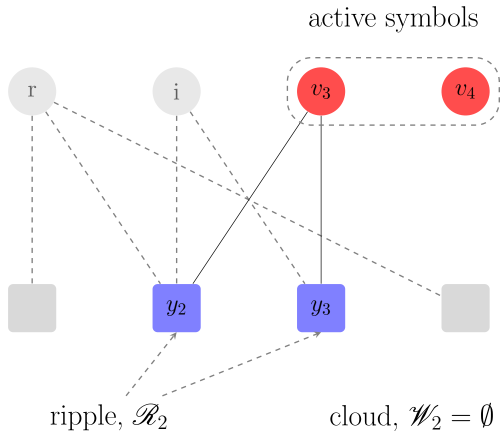


Fig. 4.8 Triangulation procedure example, $u = 2$.

3. Transition from $u = 2$ to $u = 1$. We can see in Figure 4.8 how the ripple is not empty, in fact, $\mathtt{r}_2 = 2$. Source symbol v_3 is marked as resolvable and all its attached edges are removed. Nodes y_2 and y_3 leave the graph because their reduced degree becomes zero (see Figure 4.9).

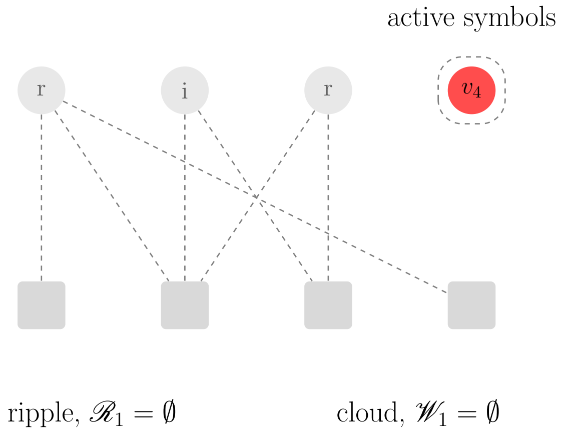

active symbols
r
i
r
v_4
ripple, $\mathscr{R}_1 = \emptyset$
cloud, $\mathscr{W}_1 = \emptyset$

Fig. 4.9 Triangulation procedure example, $u = 1$.

4. Transition from $u = 1$ to $u = 0$. In Figure 4.9 we can see how the ripple and cloud are now empty. Hence, an inactivation takes place: node v_4 is marked as inactive and the triangulation procedure ends.

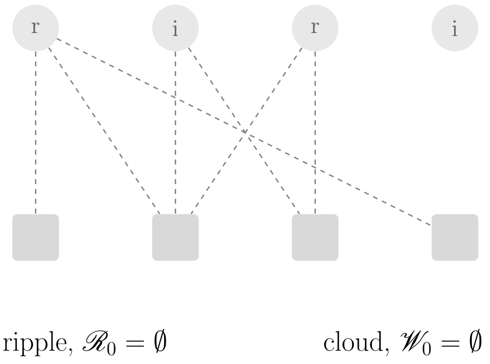

r
i
r
i
ripple, $\mathscr{R}_0 = \emptyset$
cloud, $\mathscr{W}_0 = \emptyset$

Fig. 4.10 Triangulation procedure example, $u = 0$.

For illustration we also show the effect of triangulation on the generator matrix $\tilde{\mathbf{G}}$ in Figure 4.11. Concretely, on the left hand side we can see matrix $\tilde{\mathbf{G}}$ before the triangulation procedure starts, and on the right hand side we can see matrix $\tilde{\mathbf{G}}$ after the triangulation procedure ends. We can see that after triangulation the upper left corner of matrix $\tilde{\mathbf{G}}$ has an upper triangular shape. We can also see that triangulation reorders the rows and columns of matrix $\tilde{\mathbf{G}}$ in order to create an upper diagonal matrix. Concretely, the first row corresponds to the first input symbol that was marked as resolvable, v_1. The second row corresponds to the second input symbol that was marked as resolvable, v_3. The last row corresponds to the first inactivated input symbol, v_2, and the second to last (third) row to the second symbol that was marked inactive v_4. In the following we will stick to a bipartite graph representation of the triangulation procedure, since it allows us to ignore the row and column reordering, making inactivation conceptually simpler.

$$\begin{array}{c|cccc} & y_1 & y_2 & y_3 & y_4 \\ \hline v_1 & 1 & 1 & 0 & 1 \\ v_2 & 0 & 1 & 1 & 0 \\ v_3 & 0 & 1 & 1 & 0 \\ v_4 & 0 & 0 & 0 & 0 \end{array}$$

(a) Before triangulation

$$\begin{array}{c|cccc} & y_1 & y_2 & y_3 & y_4 \\ \hline v_1 & 1 & 1 & 0 & 1 \\ v_3 & 0 & 1 & 1 & 0 \\ v_4 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 1 & 1 & 0 \end{array}$$

(b) After triangulation

Fig. 4.11 Generator matrix $\tilde{\mathbf{G}}$ before and after the triangulation procedure.

.

4.1.1 Inactivation Strategies

In this thesis we will focus mostly on a specific inactivation strategy, namely, random inactivation. This inactivation strategy is chosen in order to render the analysis trackable. However, other inactivation strategies exist that lead to a lower number of inactivations, and, therefore, to a decreased decoding complexity. In this Section we will give a short overview of the different inactivation strategies that can be found in literature.

An inactivation consists simply of marking one of the active input symbols as inactive. Therefore, when performing an inactivation the decoder will be presented with as many choices as active symbols are present at that decoding stage. For illustration, we provide a reduced decoding graph in Figure 4.12. The reduced decoding graph represents an LT code with $k = 9$ source nodes and $m = 10$ output nodes. It can be observed how source nodes v_6 and v_7 have been previously marked as resolvable, whereas source nodes v_8 and v_9 have been previously marked as inactive. Source nodes v_1, v_2, v_3, v_4 and v_5 are still active. We can also see how 3 output symbols have left the reduced graph, since their reduced degree is zero. Output symbols y_1, y_2, ..., y_7 are still in the reduced graph.

In the following we will present several algorithms that correspond to different inactivation strategies.

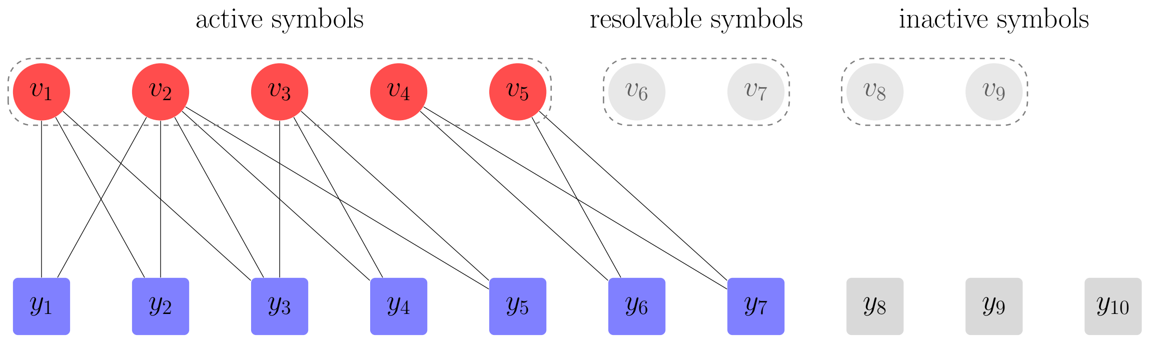


Fig. 4.12 Example of decoding graph of an LT code under inactivation decoding

Algorithm 3 (Random inactivation)**.** *One of the active input symbols, v, is chosen uniformly at random. This input symbol is marked as inactive and leaves the reduced graph*[1]*.*

Note that random inactivation does not depend on the bipartite graph formed by the active symbols. The main advantages of this strategy are its simplicity, and the fact that it renders the analysis on inactivation decoding trackable. If random inactivation is applied to the decoding graph in Figure 4.12 one of the five active input symbols is chosen at uniformly at random.

Algorithm 4 (Maximum reduced degree inactivation)**.** *The decoder marks as inactive the source symbol with maximum reduced degree. In case there are several source symbols with the same reduced degree*[2]*, one of the source symbols with maximum reduced degree is selected uniformly at random and it is marked as inactive.*

Maximum reduced degree inactivation aims at making the bipartite graph as sparse as possible by inactivating the input symbol with the most edges attached. For illustration, we will apply maximum reduced degree inactivation to the decoding graph in Figure 4.12. We can see how among the four input symbols, v_2 has reduced degree 5, v_1 and v_3 have reduced degree 3, and v_4 and v_5 have reduced degree 2. Hence, the decoder will inactivate v_2.

In order to introduce the next inactivation strategy we first need to introduce a definition

[1] This algorithm is a sub-algorithm of Algorithm 2. It is provided for the sake of completeness.

[2] As for output symbols, the reduced degree of an input symbol is its degree in the reduced graph.

Definition 8 (Accumulated reduced degree)**.** *The accumulated reduced degree of an output symbol is defined as the sum of the reduced degrees of all its neighbors (input symbols). Formally, for a output symbol of degree d, and denoting its neighbors as $v_1, v_2, \ldots, v_d$, its accumulated reduced degree is defined as*

$$\sum_{i=1}^{d} \deg_r(v_i).$$

Making use of this definition we can introduce the next inactivation strategy.

Algorithm 5 (Maximum accumulated reduced degree inactivation)**.** *The decoder selects the output symbol with minimum reduced degree in the reduced graph. In case there are several output symbols with the same minimum reduced degree, the decoder computes the accumulated reduced degree of each of them and selects the one with maximum accumulated reduced degree. In case there are several output symbols with same minimum reduced degree and same maximum accumulated reduced degree, one of them is chosen at random. The decoder then marks as inactive one of the neighbors of the selected output symbol.*

The rational behind maximum accumulated reduced degree inactivation is trying to make an input symbol enter the ripple as soon as possible. Let us also apply maximum accumulated weight inactivation to the decoding graph in Figure 4.12. It is easy to see how we have 6 output symbols of minimum degree 2, y_1, y_2, y_4, y_5, y_6 and y_7, with respective accumulated weights 8, 8, 8, 8, 4 and 4. Thus the decoder selects one output symbol at random among those with minimum reduced degree and maximum accumulated reduced degree,i.e., among y_1, y_2, y_4 and y_5. Let us assume that y_1 is selected. Next, one of its two neighbors is selected at random, for example v_1 and is inactivated. As a consequence y_1 and y_2 enter the ripple.

In order to introduce the last inactivation strategy, it is necessary to introduce some concepts dealing with graphs. This concepts are valid for generic graphs (there is no implicit assumption of the graph being a bipartite graph).

Definition 9 (Path)**.** *In a graph, a path is a sequence of edges that connects a sequences of vertices (nodes).*

Definition 10 (Connected component)**.** *A connected component of a graph is a subgraph in which any two vertices (nodes) are connected to each other by a path, and in which no vertex (node) is connected to any vertices outside the connected component.*

Commonly, connected components are referred simply as components. Note that if two nodes, A and B belong to the same component there exists at least one path to go from A to B. This does not imply that A and B are neighbors, since the path can be composed of any number of edges. However, if A and B do not belong to the same component there is no path to go from A to B.

Algorithm 6 (Maximum component inactivation)**.**

- *The decoder searches for all degree 2 output symbols.*
- *If there are no degree 2 output symbols the decoder inactivates one input symbol at random.*
- *Otherwise (if there are 1 or more output symbols of degree 2.)*
  - *The decoder computes the unipartite graph induced by the degree 2 output symbols on the input symbols so that*
    - *each vertex in this graph corresponds to a degree 2 output symbol*
    - *two vertices of the induced unipartite graph are only connected to each other if the corresponding degree 2 output symbols have a neighbor (input symbol) in common*
  - *The decoder searches the components in the unipartite graph and computes its size (number of nodes).*
  - *The decoder inactivates one input symbol that is connected to the maximum component (that of maximum size) in the unipartite graph.*

The rational behind maximum component inactivation is that when we inactivate any of the input symbols connected to a component, in the subsequent steps of inactivation decoding all the input symbols connected to the component are marked as resolvable, and, thus, no inactivation happens.

Let us apply maximum component inactivation to the decoding graph in Figure 4.12. The degree 2 output symbols induce the graph shown in Figure 4.13. We can observe how there are two connected components. The first is formed by y_1, y_2, y_4, y_5 and the second by y_6 and y_7. The decoder hence marks as inactive one input symbol connected to the largest component. Hence, the decoder inactivates either, v_1, v_2 or v_3 (see Figure 4.12).

In practice different inactivation strategies lead to a different number of inactivations. Among the different inactivation strategies presented in this section, maximum

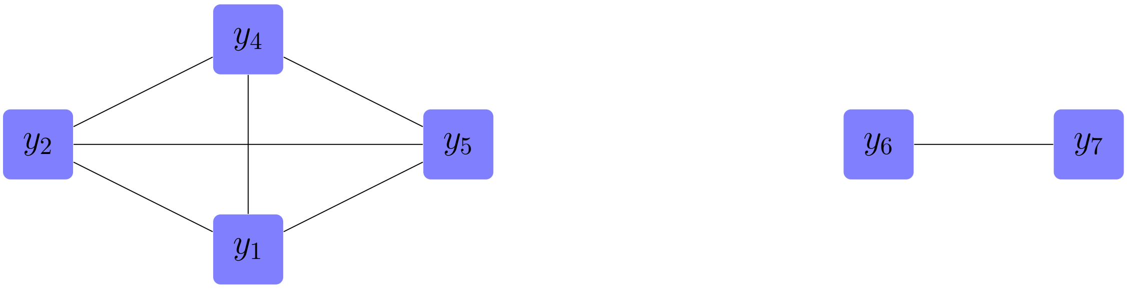


Fig. 4.13 Connected components of the decoding example.

component inactivation usually provides the least number of inactivations, followed closely by maximum accumulated weight inactivation, maximum weight inactivation and at last random inactivation. A detailed comparison of the performance of the different inactivation strategies for R10 Raptor codes can be found in Appendix A.

4.2 Analysis under Random Inactivation

In the following sections we present novel finite length analysis methods for LT codes under inactivation decoding. The goal of these methods is obtaining the number of input symbols that are inactivated after triangulation is over (or an estimation thereof).

4.2.1 First Order Finite Length Analysis

In [52, 55, 54] the iterative decoder of LT codes was analyzed using a dynamic programming approach. This analysis models the iterative decoder as a finite state machine and it can be used to derive the probability of decoding failure (under iterative decoding). In this section we extend the analysis of the iterative decoder performed in [52, 55, 54] to the inactivation decoder. The analysis we present in this section is similar to the analysis of batched sparse codes under inactivation decoding presented in [67].

As in [52, 55, 54], we model the decoder as a finite state machine with state

$$\mathtt{S}_u := (\mathtt{W}_u, \mathtt{R}_u)$$

where we recall, $\mathtt{W}_u$ and $\mathtt{R}_u$ are respectively the number of output symbols in the cloud and ripple when u output symbols are still active. In this section a recursion is derived that allows to obtain $\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_{u-1}, \mathtt{r}_{u-1})\}$ as a function of $\Pr\{\mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u)\}$.

Let us first analyze how the ripple and cloud change in the transition from u to $u-1$ active source symbols. In the transition exactly one active source symbol is marked as either resolvable or inactive and all its attached edges are removed. Whenever edges are erased in the graph the degree of one or more output symbols gets reduced. Consequently, some of the cloud symbols may enter the ripple and some of the ripple symbols may become of reduced degree zero and leave the reduced graph. We first focus on the symbols that leave the cloud and enter the ripple in the transition given that $\mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u)$. Since in an LT code the neighbors of all output symbols are selected independently and uniformly at random, in a transition each output symbol will leave the cloud and enter the ripple independently from other output symbols. Thus, the number of cloud symbols which leave $\mathscr{W}_u$ and enter $\mathscr{R}_{u-1}$ is binomially distributed with parameters $\mathtt{w}_u$ and p_u, being p_u the probability of a symbol leaving $\mathscr{W}_u$ to enter $\mathscr{R}_{u-1}$. Using Bayes' theorem p_u can be written as:

$$p_u := \Pr\{y \in \mathscr{R}_{u-1} | y \in \mathscr{W}_u\} = \frac{\Pr\{y \in \mathscr{R}_{u-1}\,,\, y \in \mathscr{W}_u\}}{\Pr\{y \in \mathscr{W}_u\}}. \tag{4.1}$$

Let us first consider the numerator of (4.1) assuming that output symbol y has (original) degree d:

$$\Pr\{y \in \mathscr{R}_{u-1}\,,\, y \in \mathscr{W}_u | \deg(y) = d\}.$$

This corresponds to the probability that

- one of the d edges of output symbol y is connected to the symbol being marked as inactive or resolvable at the transition,
- another edge is connected to one of the $u-1$ active symbols after the transition,
- the remaining $d-2$ edges connected to the $k-u$ not active input symbols (inactive or resolvable).

In other words, the symbol must have *reduced* degree 2 *before* the transition and *reduced* degree 1 *after* the transition.

Proposition 2. *The probability that a symbol y belongs to the cloud at step u and enters the ripple at step $u-1$, given its original degree d is given by*

$$\Pr\{y \in \mathscr{R}_{u-1}\,,\, y \in \mathscr{W}_u | \deg(y) = d\} = \begin{cases} \frac{d}{k}(d-1)\frac{u-1}{k-1}\frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}} & \text{if } d \geq 2 \\ 0 & \text{if } d < 2 \end{cases}. \tag{4.2}$$

Proof. First, the probability that one edge is connected to the symbol being marked as inactive or resolvable at the transition is $1/k$, and there are d distinct choices for the edge connected to it. This accounts for the term d/k in (4.2).

Second, there are $d-1$ choices for the edge going to the $u-1$ active symbols after the transition, and the probability of an edge being connected to the set of $u-1$ active symbols is $(u-1)/(k-1)$. This is reflected in the term $(d-1)(u-1)/(k-1)$ in (4.2).

Finally, the last term corresponds to the probability of having exactly $d-2$ edges going to the $k-u$ not active input symbols:

$$\frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}}. \qquad \square$$

If the conditioning on d in (4.2) is removed we obtain

$$\Pr\{y \in \mathscr{R}_{u-1}\,,\, y \in \mathscr{W}_u\} = \sum_{d=2}^{d_{\max}} \Omega_d \frac{d}{k}(d-1)\frac{u-1}{k-1}\frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}}. \tag{4.3}$$

The denominator of (4.1) is given by the probability that the randomly chosen output symbol y is in the cloud when u input symbols are still active. This is equivalent to the probability of not being in the ripple or having reduced degree zero (all edges are going to symbols marked as inactive or resolvable) as provided by the following Proposition.

Proposition 3. *The probability that the randomly chosen output symbol y is in the cloud when u input symbols are still active corresponds to*

$$\Pr\{y \in \mathscr{W}_u\} = 1 - \sum_{d=1}^{d_{\max}} \Omega_d \left[u\frac{\binom{k-u}{d-1}}{\binom{k}{d}} + \frac{\binom{k-u}{d}}{\binom{k}{d}} \right]. \tag{4.4}$$

Proof. The probability of y not being in the cloud is given by the probability of y having reduced degree 0 or being in the ripple. Since the two events are mutually exclusive, we can compute such probability as the sum of two probabilities, the probability of y being in the ripple (i.e., having reduced degree 1) and the probability of y having reduced degree 0.

We will first focus on the probability of y being in the ripple. Let us assume y is of degree d. The probability that y has reduced degree 1 equals the probability of y having exactly one neighbor among the u active source symbols and the remaining

$d-1$ neighbors among the $k-u$ non-active ones. This is given by

$$d\,\frac{u}{k}\,\frac{\binom{k-u}{d-1}}{\binom{k-1}{d-1}} = u\frac{\binom{k-u}{d-1}}{\binom{k}{d}}$$

that corresponds to the first term in (4.4).

The probability of y having reduced degree 0 is the probability that all d neighbors of y are in the $k-u$ non-active symbols. This leads to the term

$$\frac{\binom{k-u}{d}}{\binom{k}{d}}$$

in (4.4). □

The probability p_u can be finally obtained through (4.1), making use of (4.3) and of (4.4) and corresponds to:

$$p_u = \frac{\sum_{d=2}^{\min(d_{\max},k-u+2)} \Omega_d\, d\,(d-1) \frac{1}{k}\frac{u-1}{k-1}\frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}}}{1 - \sum_{d=1}^{\min(d_{\max},k-u+1)} \Omega_d\, u \frac{\binom{k-u}{d-1}}{\binom{k}{d} d} - \sum_{d=1}^{\min(d_{\max},k-u)} \Omega_d \frac{\binom{k-u}{d}}{\binom{k}{d}}}$$

where the upper limits of the summations have been adjusted to take into account that an output symbol cannot have more than d edges going to a set of d input symbols.

Let us now focus on the number of symbols leaving the ripple during the transition from u to $u-1$ active symbols, which we shall denote by $\mathtt{a}_u$. We denote by $\mathtt{A}_u$ the random variable associated with $\mathtt{a}_u$. We distinguish two cases. In the first case, the ripple is not empty and no inactivation takes place. Hence, an output symbol y is chosen at random from the ripple and its only neighbor v is marked as resolvable and it is removed from the graph. Any other output symbol in the ripple which is connected to the input symbol v leaves the ripple during the transition. Hence, for $\mathtt{r}_u > 0$ we have

$$\Pr\{\mathtt{A}_u = \mathtt{a}_u | \mathtt{R}_u = \mathtt{r}_u\} = \binom{\mathtt{r}_u - 1}{\mathtt{a}_u - 1} \left(\frac{1}{u}\right)^{\mathtt{a}_u - 1} \left(1 - \frac{1}{u}\right)^{\mathtt{r}_u - \mathtt{a}_u}$$

with $1 \leq \mathtt{a}_u \leq \mathtt{r}_u$.

In the second case, the ripple is empty ($\mathtt{r}_u = 0$). Since no output symbols can leave the ripple, we have

$$\Pr\{\mathtt{A}_u = \mathtt{a}_u | \mathtt{R}_u = 0\} = \begin{cases} 1 & \text{if } \mathtt{a}_u = 0 \\ 0 & \text{if } \mathtt{a}_u \neq 0 \end{cases}.$$

Now we are in the position to derive the transition probability

$$\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_{u-1}, \mathtt{r}_{u-1}) | \mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u)\}.$$

Let us recall that, by definition, $\mathtt{b}_u$ denotes the variation of number of cloud elements in the transition from u to $u-1$ active symbols. Formally,

$$\mathtt{b}_u := \mathtt{w}_u - \mathtt{w}_{u-1}.$$

Let us also observe that the variation of the ripple size is subject to the following equilibrium constraint

$$\mathtt{a}_u - \mathtt{b}_u = \mathtt{r}_u - \mathtt{r}_{u-1}$$

which follows from the definition $\mathtt{a}_u$ and $\mathtt{b}_u$. The transition probability is given by:

$$\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_u - \mathtt{b}_u, \mathtt{r}_u - \mathtt{a}_u + \mathtt{b}_u) | \mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u)\} = \\ \binom{\mathtt{w}_u}{\mathtt{b}_u} {p_u}^{\mathtt{b}_u} (1 - p_u)^{\mathtt{w}_u - \mathtt{b}_u} \binom{\mathtt{r}_u - 1}{\mathtt{a}_u - 1} \left(\frac{1}{u}\right)^{\mathtt{a}_u - 1} \left(1 - \frac{1}{u}\right)^{\mathtt{r}_u - \mathtt{a}_u} \tag{4.5}$$

for $\mathtt{r}_u > 0$, while for $\mathtt{r}_u = 0$ we have

$$\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_u - \mathtt{b}_u, \mathtt{b}_u) | \mathtt{S}_u = (\mathtt{w}_u, 0)\} = \binom{\mathtt{w}_u}{\mathtt{b}_u} {p_u}^{\mathtt{b}_u} (1 - p_u)^{\mathtt{w}_u - \mathtt{b}_u}. \tag{4.6}$$

Finally, the probability of the decoder being in state $\mathtt{S}_{u-1} = (\mathtt{w}_{u-1}, \mathtt{r}_{u-1})$ can be computed in a recursive manner via (4.5), (4.6). The decoder state is initialized as

$$\Pr\{\mathtt{S}_k = (\mathtt{w}_k, \mathtt{r}_k)\} = \binom{m}{\mathtt{r}_k} \Omega_1^{\mathtt{r}_k} (1 - \Omega_1)^{\mathtt{w}_k}$$

for all non-negative $\mathtt{w}_k, \mathtt{r}_k$ such that $\mathtt{w}_k + \mathtt{r}_k = m$ where m is the number of output symbols.

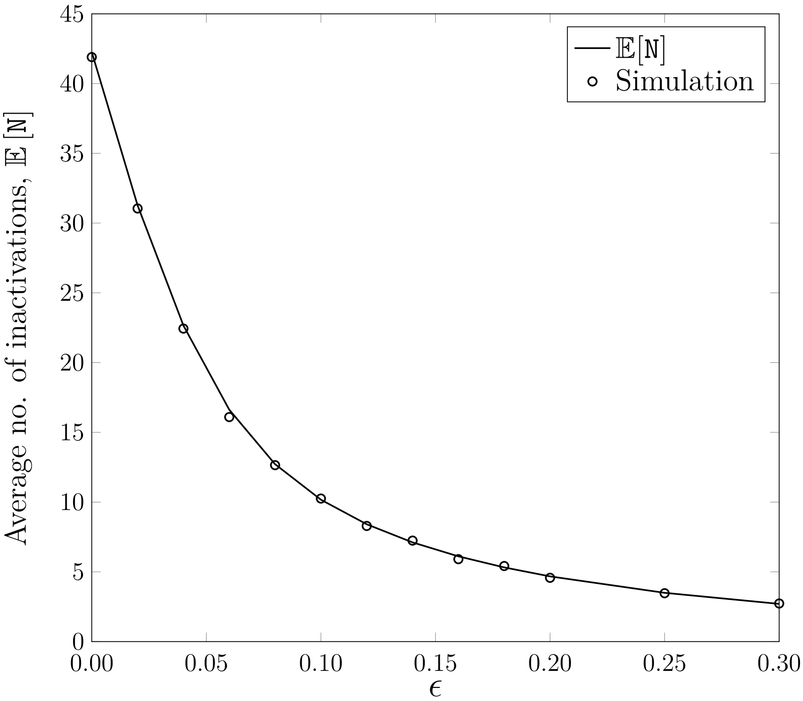


Fig. 4.14 Average number of inactivations vs. relative overhead ϵ for an LT code with $k = 1000$ and with degree distribution Ω^{R10}.

Let us denote by $\mathtt{N}$ the random variable that corresponds to the cumulative number of inactivations after the k steps. The expected value of $\mathtt{N}$ is given by

$$\mathbb{E}[\mathtt{N}] = \sum_{u=1}^{k} \sum_{\mathtt{w}_u} \Pr\{\mathtt{S}_u = (\mathtt{w}_u, 0)\}. \tag{4.7}$$

Figure 4.14 shows the expected number of inactivations for an LT code with $k = 1000$ and the output degree distribution used in standardized Raptor codes, Ω^{R10}. The chart compares the average number of inactivations obtained through Monte Carlo simulation and by (4.7). It can be observed how there is a tight match between the analysis and the simulation results.

4.2.2 Complete Finite Length Analysis

The analysis presented in Section 4.2.1 is able to provide the expected number of inactivations (first moment). In this section we shall see that the model can be easily modified to obtain also the complete probability distribution of the number of inactivations. For this purpose, we first need to include in the state definition the number of inactive input symbols. Hence the state is given by

$$\mathtt{S}_u = (\mathtt{W}_u, \mathtt{R}_u, \mathtt{N}_u)$$

with $\mathtt{N}_u$ being the random variable that corresponds to the number of inactivations at step u. Again, we proceed by deriving a recursion that allows deriving $\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_{u-1}, \mathtt{r}_{u-1}, \mathtt{n}_{u-1})\}$ as a function of $\Pr\{\mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u, \mathtt{n}_u)\}$. Let us first look at the transition from u to $u-1$ active symbols when $\mathtt{r}_u \geq 1$, that is, when no inactivation takes place. In this case the number of inactivations stays the same and we have $\mathtt{n}_{u-1} = \mathtt{n}_u$. Therefore, we have

$$\begin{aligned}\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_u - \mathtt{b}_u, \mathtt{r}_u - \mathtt{a}_u + \mathtt{b}_u, \mathtt{n}_u) | \mathtt{S}_u = (\mathtt{w}_u, \mathtt{r}_u, \mathtt{n}_u)\} = \\ \binom{\mathtt{w}_u}{\mathtt{b}_u} {p_u}^{b_u} (1-p_u)^{\mathtt{w}_u - \mathtt{b}_u} \binom{\mathtt{r}_u - 1}{\mathtt{a}_u - 1} \left(\frac{1}{u}\right)^{\mathtt{a}_u - 1} \left(1 - \frac{1}{u}\right)^{\mathtt{r}_u - \mathtt{a}_u}.\end{aligned} \tag{4.8}$$

Let us now look at the transition from u to $u-1$ active symbols when $\mathtt{r}_u = 0$, that is, when an inactivation takes place. In this case the number of inactivations is increased by one yielding

$$\Pr\{\mathtt{S}_{u-1} = (\mathtt{w}_u - \mathtt{b}_u, \mathtt{b}_u, \mathtt{n}_u + 1) | \mathtt{S}_u = (\mathtt{w}_u, 0, \mathtt{n}_u)\} = \binom{\mathtt{w}_u}{\mathtt{b}_u} {p_u}^{\mathtt{b}_u} (1-p_u)^{\mathtt{w}_u - \mathtt{b}_u}. \tag{4.9}$$

The probability of the decoder being in state $\mathtt{S}_{u-1} = (\mathtt{w}_{u-1}, \mathtt{r}_{u-1}, \mathtt{n}_{u-1})$ can be computed recursively via (4.8), (4.9) starting with the initial condition

$$\Pr\{\mathtt{S}_k = (\mathtt{w}_k, \mathtt{r}_k, \mathtt{n}_u)\} = \binom{m}{\mathtt{r}} \Omega_1^{\mathtt{r}} (1 - \Omega_1)^{\mathtt{w}_k}$$

for all non-negative $\mathtt{w}_k, \mathtt{r}_k$ such that $\mathtt{w}_k + \mathtt{r}_k = m$ and $\mathtt{n}_k = 0$.

The distribution of the number of inactivations needed to complete the decoding process is finally given by

$$f_{\mathtt{N}}(\mathtt{n}) = \sum_{\mathtt{w}_0} \sum_{\mathtt{r}_0} \Pr\{\mathtt{S}_0 = (\mathtt{w}_0, \mathtt{r}_0, \mathtt{n})\}. \tag{4.10}$$

From (4.10) we may obtain the cumulative distribution $F_{\mathtt{N}}(\mathtt{n})$ which corresponds to the probability of performing at most $\mathtt{n}$ inactivations during the decoding process. The cumulative distribution of the number of inactivations has practical implications. Let us assume the fountain decoder runs on a platform with limited computational

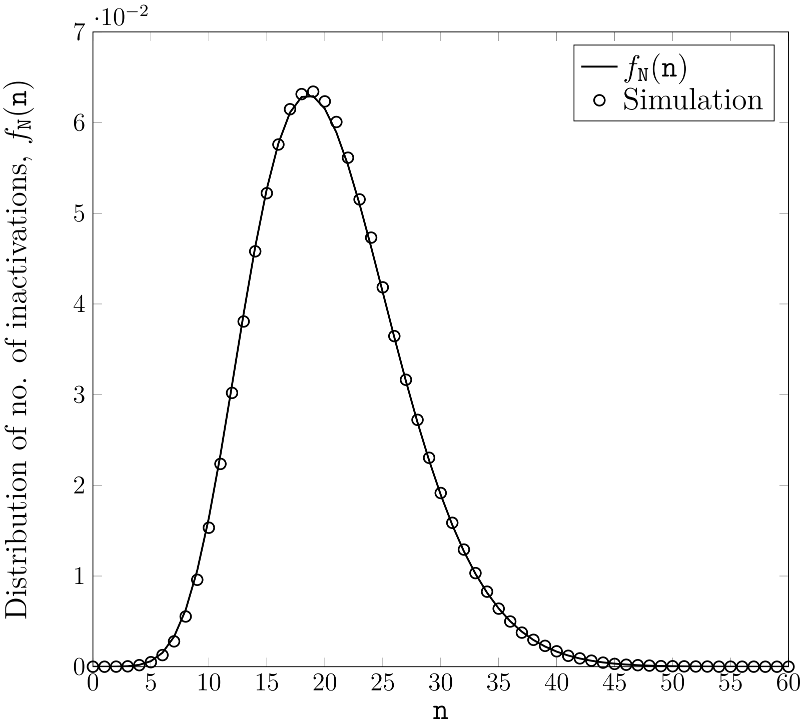


Fig. 4.15 Distribution of the number of inactivations for an LT code with $k = 300$ and degree distribution Ω^{R10} given in (3.10).

capability. For example, the decoder may be able to perform a maximum number of inactivations (recall that the complexity of inactivation decoding is cubic in the number of inactivations, $\mathtt{n}$). Suppose the maximum number of inactivations that the decoder can handle is $\mathtt{n}^*$. For such a decoder, the probability of decoding failure will be lower bounded by $1 - F_{\mathtt{N}}(\mathtt{n}^*)^3$.

Figure 4.15 shows the distribution of the number of inactivations, for an LT code with degree distribution Ω^{R10} given in (3.10) and source block size $k = 500$. The chart shows the distribution of the number of inactivations obtained through both Monte Carlo simulation and by (4.10). Again, we can observe how there is a very tight match between the analysis and the simulation results.

4.2.3 Binomial Approximation

In Sections 4.2.1 and 4.2.2 we have derived recursive methods that can compute the expected number of inactivations and the distribution of the number of inactivations

[3] The probability of decoding failure is actually higher than $1 - F_{\mathtt{N}}(\mathtt{n}^*)$ since the system of equations to be solved in the Gaussian elimination (GE) step of inactivation decoding might be rank deficient.

needed to complete the decoding process. The disadvantage of these methods is that their evaluation is computationally complex for large k. In this section we propose another recursive method that can provide a reasonably accurate estimation of the number of inactivations with lower computational complexity.

Let us start by introducing the following definition.

Definition 11 (Reduced degree-d set)**.** *The reduced degree-d set is defined as the set of output symbols of reduced degree d, and we denote it by $\mathscr{Z}^{(d)}$.*

We shall denote the cardinality of $\mathscr{Z}^{(d)}$ by $\mathtt{z}^{(d)}$ and its associated random variable by $\mathtt{Z}^{(d)}$. Obviously, the set $\mathscr{Z}^{(1)}$ corresponds to the ripple. Moreover, the cloud $\mathscr{W}$ corresponds to the union of the sets of output symbols of reduced degree higher than 1, that is,

$$\mathscr{W} = \bigcup_{d=2}^{d_{\max}} \mathscr{Z}^{(d)}.$$

Furthermore, since the sets $\mathscr{Z}^{(d)}$ are obviously disjoint, we have

$$\mathtt{W} = \sum_{d=2}^{d_{\max}} \mathtt{Z}^{(d)}.$$

As we did in the previous section, we shall add a temporal dimension through subscript u, where u corresponds to the number of active input symbols in the graph. Thus, $\mathscr{Z}_u^{(d)}$ is the set of reduced degree d output symbols when u input symbols are still active. Furthermore, $\mathtt{Z}_u^{(d)}$ and $\mathtt{z}_u^{(d)}$ are respectively the random variable associated to the number of reduced degree d output symbols when u input symbols are still active and its realization. The triangulation process can be modelled by means of a finite state machine with state

$$\mathtt{S}_u := \left(\mathtt{Z}_u^{(1)}, \mathtt{Z}_u^{(2)}, \ldots, \mathtt{Z}_u^{(d_{\max})}\right).$$

Let us analyze next how the change of the number of reduced degree d output symbols in the transition from u to $u-1$ active input symbols. We shall consider first a randomly chosen output symbol y with reduced degree $d \geq 2$. Denoting by $p_u^{(d)}$ the probability that the degree of y decreases to $d-1$ after the transition from u to $u-1$, that is,

$$p_u^{(d)} := \Pr\{y \in \mathscr{X}_{u-1}^{(d-1)} | y \in \mathscr{X}_u^{(d)}\}$$

we have,

Proposition 4. *The probability that a randomly chosen output symbol y, that has reduced degree $d \geq 2$ when u input symbols are active, has reduced degree $d-1$ when $u-1$ input symbols are active is*

$$p_u^{(d)} = \frac{d}{u}.$$

Proof. Before the transition we have u active input symbols and output symbol y has exactly d neighbors among the u active input symbols. In the transition from u to $u-1$ active symbols, 1 input symbol is selected at random and marked as either resolvable or inactive. The probability that the degree of y gets reduced corresponds to the probability that one of its d neighbors is marked as resolvable or inactive. □

Since all output symbols choose their neighbors independently, we have that the random variable associated with the number of output symbols of reduced degree d that become of reduced degree $d-1$ in the transition from u to $u-1$ output symbols, $B_u^{(d)}$, conditioned to $\mathsf{Z}_u^{(d)} = \mathsf{z}_u^{(d)}$, is binomially distributed with parameters $\mathsf{z}_u^{(d)}$ and $p_u^{(d)} = d/u$.

Since output symbols select their neighbors without replacement, they cannot have two edges going to the same input symbols. Thus, we have,

$$\mathsf{Z}_{u-1}^{(d)} = \mathsf{Z}_u^{(d)} + B_u^{(d+1)} - B_u^{(d)} \tag{4.11}$$

Using a dynamical programming approach similar to the one in Section 4.2.1 it is possible to derive a recursion to determine the decoder state probability, which also provides the expected number of inactivations. However, such a recursion would be much more complex to evaluate than the one in Section 4.2.1 because the number of possible states is now much larger. Nevertheless, the complexity can be dramatically reduced by introducing a simplifying assumption. Concretely, the method presented in this section is based on the assumption that $\mathsf{Z}_u^{(d)}$ can be approximated by a binomially distributed random variable with parameters m and $\xi_u^{(d)}$. Formally, we assume that the distribution of the decoder state at step u can be approximated as a product of binomial distributions,

$$\Pr\{\mathsf{S}_u = \mathbf{z}_u\} \approx \prod_{d=1}^{d_{\max}} \binom{m}{\mathsf{z}_u^{(d)}} \left(\xi_u^{(d)}\right)^{\mathsf{z}_u^{(d)}} \left(1-\xi_u^{(d)}\right)^{m-\mathsf{z}_u^{(d)}} .$$

where

$$\mathbf{z}_u = \left(\mathsf{z}_u^{(1)}, \mathsf{z}_u^{(2)}, \ldots, \mathsf{z}_u^{(d_{\max})}\right)$$

This binomial distribution assumption is made for the sake of simplicity but it was shown to be reasonably accurate through Monte Carlo simulations. This assumption greatly simplifies the finite state machine, since the finite length analysis now reduces to deriving a recursion to obtain $\xi_u^{(d)}$.

In order to derive a recursion to compute $\xi_u^{(d)}$, we shall distinguish two cases. First, we consider the output symbols of reduced degree $d \geq 2$. From Proposition 4, for $d \geq 2$ we have that $B_u^{(d)}$ conditioned to $\mathsf{Z}_u^{(d)} = \mathsf{z}_u^{(d)}$ is a binomial random variable with parameters $\mathsf{z}_u^{(d)}$ and $p_u^{(d)} = d/u$. Thus, we have

$$\mathbb{E}\left[\mathbb{E}\left[B_u^{(d)} | \mathsf{Z}_u^{(d)}\right]\right] = \frac{d}{u}\,\mathbb{E}\left[\mathsf{Z}_u^{(d)}\right]. \tag{4.12}$$

If we now take the expectation on both sides of (4.11), we have

$$\mathbb{E}\left[\mathsf{Z}_{u-1}^{(d)}\right] = \mathbb{E}\left[\mathsf{Z}_u^{(d)}\right] + \mathbb{E}\left[B_u^{(d+1)}\right] - \mathbb{E}\left[B_u^{(d)}\right]. \tag{4.13}$$

Substituting (4.12) in (4.13) , and we can write

$$\mathbb{E}\left[\mathsf{Z}_{u-1}^{(d)}\right] = \mathbb{E}\left[\mathsf{Z}_u^{(d)}\right] + \frac{d+1}{u}\,\mathbb{E}\left[\mathsf{Z}_u^{(d+1)}\right] - \frac{d}{u}\,\mathbb{E}\left[\mathsf{Z}_u^{(d)}\right].$$

If we now make use of our assumption and consider

$$\mathbb{E}\left[\mathsf{Z}_{u-1}^{(d)}\right] = m\,\xi_u^{(d)}$$

we can write

$$\xi_{u-1}^{(d)} = \left(1 - \frac{d}{u}\right)\xi_u^{(d)} + \frac{d+1}{u}\xi_u^{(d+1)},$$

for $d \geq 2$.

We shall now consider the output symbols of reduced degree 1. We are interested in $B_u^{(1)}$, the random variable associated to the output symbols of reduced degree $d = 1$

that become of reduced degree 0 in the transition from u to $u-1$ active input symbols. If we assume that the ripple is not empty, $\mathtt{z}_u^{(1)} \geq 1$, an output symbol y is chosen at random from the ripple and its only neighbor v is marked as resolvable and it is removed from the graph. Moreover, any other output symbol in the ripple being connected to input symbol v also leaves the ripple during the transition. Thus, for $\mathtt{z}_u^{(1)} \geq 1$ we have

$$\mathbb{E}\left[B_u^{(1)} | \mathtt{Z}_u^{(1)} = \mathtt{z}_u^{(1)} \geq 1\right] = 1 + \frac{1}{u}\left(\mathtt{z}_u^{(1)} - 1\right) = 1 - \frac{1}{u} + \frac{1}{u}\mathtt{z}_u^{(1)} \tag{4.14}$$

whereas when the ripple is empty, $\mathtt{z}_u^{(1)} = 0$, we have

$$\mathbb{E}\left[B_u^{(1)} | \mathtt{Z}_u^{(1)} = 0\right] = 0. \tag{4.15}$$

If we put together (4.14) and (4.15) we have

$$\begin{aligned}
\mathbb{E}\left[B_u^{(1)}\right] &= \left(1 - \frac{1}{u}\right) \Pr\{\mathtt{Z}_u^{(1)} \geq 1\} + \frac{1}{u} \sum_{\mathtt{z}_u^{(1)}=1}^{m} \mathtt{z}_u^{(1)} \Pr\{\mathtt{Z}_u^{(1)} = \mathtt{z}_u^{(1)}\} \\
&= \left(1 - \frac{1}{u}\right)\left(1 - \Pr\{\mathtt{Z}_u^{(1)} = 0\}\right) + \frac{1}{u}\mathbb{E}\left[\mathtt{Z}_u^{(1)}\right] \\
&= \left(1 - \frac{1}{u}\right)\left(1 - \left(1 - \xi_u^{(1)}\right)^m\right) + \frac{1}{u}\, m\, \xi_u^{(1)} \qquad (4.16)
\end{aligned}$$

If we now replace (4.16) into (4.13) we obtain the following recursion for $\xi_u^{(1)}$,

$$\xi_{u-1}^{(1)} = \left(1 - \frac{1}{u}\right)\xi_u^{(1)} + \frac{2}{u}\xi_u^{(2)} - \frac{(1 - 1/u)\left(1 - (1 - \xi_u^{(1)})^m\right)}{m}$$

In order to initialize the finite state machine we shall assume that before triangulation starts $\mathtt{Z}_k^{(d)}$ follows a binomial distribution $\mathcal{B}(m, \Omega_d)$, i.e., we assume

$$\xi_k^{(d)} = \Omega_d.$$

The probability of an inactivation occurring in the transition from u to $u-1$ active symbols corresponds simply to the probability of the ripple being empty

$$\Pr\{\mathtt{Z}_u^{(1)} = 0\} = \left(1 - \xi_u^{(1)}\right)^m.$$

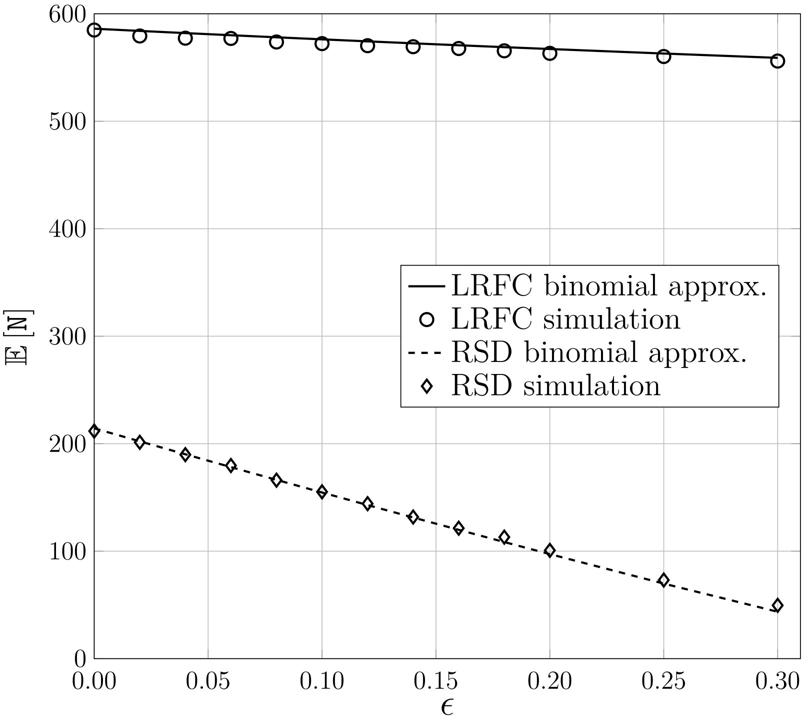


Fig. 4.16 Average number of inactivations needed to decode a linear random fountain code and a RSD for $k = 1000$ and average output degree $\bar{\Omega} = 12$. The markers represent simulation results and the lines represent the predicted number of inactivations for random inactivation using the binomial approximation.

Let us recall that $\mathtt{N}$ is the random variable associated with the cumulative number of inactivations after the k steps of triangulation, we have

$$\mathbb{E}[\mathtt{N}] = \sum_{u=1}^{k} \Pr\{\mathtt{Z}_u^{(1)} = 0\}.$$

Figure 4.16 shows the average number of inactivations needed to complete decoding for a LRFC[4] and a RSD with parameters $\varsigma = 0.09266$ and $\psi = 0.001993$, both with average output degree $\bar{\Omega} = 12$ and $k = 1000$. The figure shows results obtained by Monte Carlo simulation and also the estimation obtained under our binomial approximation. A tight match between simulation results and the estimation can be observed.

In Figure 4.17 we shows the evolution of the ripple size $\mathtt{R}_u = \mathtt{Z}_u^{(1)}$ and the cumulative number of inactivations when u input symbols are active. The output degree distribution

[4] The degree distribution of a LRFC follows a binomial distribution.

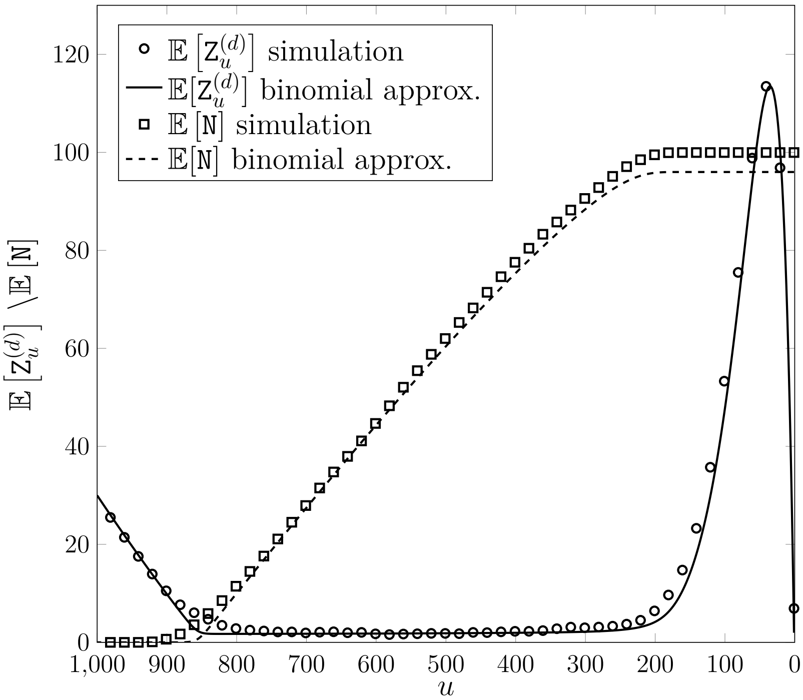


Fig. 4.17 Evolution of $\mathtt{R}_u$ and the cumulative number of inactivations with respect to the number of active input symbols u. The output degree distribution is a RSD with $k = 1000$ and $\epsilon = 0.2$. The lines represent the prediction obtained under the binomial distribution assumption. The markers represent the average obtained through Monte Carlo simulations.

is again a RSD with parameters $\varsigma = 0.09266$ and $\psi = 0.001993$ and the relative receiver overhead is $\epsilon = 0.2$. The figure shows the result of Monte Carlo simulations and the estimation obtained with our binomial approximation. It can be observed how the match between simulation results and the outcome of our approximation is tight.

4.3 Code Design

This section focuses on the design of LT codes optimized for inactivation decoding. Most of the works on LT codes assume iterative decoding rather than inactivation decoding. An exception is the work in [68] where the authors derived a analytically family of degree distributions optimized for inactivation decoding. The authors in [68] found out that in the unconstrained case, the optimal degree distribution under inactivation decoding corresponds to the ideal soliton distribution. One of the main shortcomings of this work is that there is no direct control on the average output degree of the degree distributions, which implies one has no control on the encoding complexity.

In this section we present method to design LT codes for inactivation decoding. In this chapter three different methods have been presented to obtain the number of inactivations needed to complete decoding (or an estimation thereof). Any of these three methods can be used to perform a numerical optimization of the output degree distribution which leads to a *low* number of inactivations. We will use the approximate method presented in Section 4.2.3 since its evaluation is much faster than that of the other two methods. Nevertheless, it is still feasible to use the other methods for numerical code design provided that the source block size k is not too large (up to several hundreds).

The numerical optimization algorithm we will use is simulated annealing (SA) [69], a fast meta-heuristic method for global optimization that is inspired in the process of annealing in metallurgy in which a material is cooled down slowly to obtain a crystal structure. The starting point of SA corresponds to an initial state s_{init} plus an initial temperature T_{init}. At every step a number of candidate successive states for the system are generated as a slight variation of the previous state and the temperature decreases. For high temperatures SA allows moving the system to higher energy states with some probability that becomes smaller as the temperature of the system decreases. This process is repeated until the system reaches a target energy or until a maximum number of steps are carried out. In our case the states correspond to degree distributions. The energy of a state (degree distribution) has to be defined so that it is a monotonically decreasing function of the number of inactivations $\mathbb{E}\left[\mathtt{N}\right]$ and the probability of decoding failure P_{F}.

Concretely, we consider a source block size $k = 10000$ and we set the following constraints:

- A target probability of decoding failure $P_f^* = 10^{-2}$ at $\epsilon = 0$.
- Maximum average output degree $\bar{\Omega} \leq 12$.
- Maximum output degree $d_{\max} = 150$.

The second and third constraint are introduced to limit the average and worst case encoding cost of an output symbol. In order to embed these constraints into SA the objective function to be minimized (energy) is defined as

$$\Upsilon = \mathbb{E}\left[\hat{\mathtt{N}}\right] + f_p(\underline{\mathsf{P}}_{\mathsf{F}})$$

where f_p is defined as

$$f_p(\underline{\mathsf{P}}_\mathsf{F}) = \begin{cases} 0, & \underline{\mathsf{P}}_\mathsf{F} < \mathsf{P}_\mathsf{F}{}^* \\ b\,(1 - \mathsf{P}_\mathsf{F}{}^* / \underline{\mathsf{P}}_\mathsf{F}), & \text{otherwise} \end{cases} \tag{4.17}$$

being $\mathsf{P}_\mathsf{F}{}^*$ the target probability of decoding failure, $\underline{\mathsf{P}}_\mathsf{F}$ the tight lower bound to it given in equation (3.5) and b a large positive number ($b = 1000$ was used in the example). The large b factor ensures that degree distributions which do not comply with the target probability of decoding failure are discarded.

The use of $\underline{\mathsf{P}}_\mathsf{F}$ in place of the actual P_F in the objective functions stems from the need of having a fast (though, approximate) performance estimation to be used within the SA recursion (note in fact that the evaluation of the actual P_F may present a prohibitive complexity). This allows evaluating the energy of a state (i.e., degree distribution) very quickly. Although the lower bound in eq. (3.5) may not be tight for $\epsilon = 0$, it is very tight for values of ϵ slightly larger than 0. This means that in practice we will need ϵ slightly larger than 0 to comply with our requirements.

For the sake of illustration we will perform two different optimizations. In the first one we will constrain the degree distribution to be a (truncated) RSD and in the second one we will impose no constraints to the degree distribution.

Let Ω^{RSD} be a RSD distribution. For a given maximum degree $d_{\max}$ we define the truncated RSD distribution, $\Omega^{\text{RSD}'}$, as

$$\Omega_i^{\text{RSD}'} = \begin{cases} \Omega_i^{\text{RSD}}, & i < d_{\max} \\ \sum_{j=d_{\max}}^{k} \Omega_j^{\text{RSD}}, & i = d_{\max} \\ 0, & i > d_{\max} \end{cases} .$$

Therefore, in this first optimization we aim at finding the RSD parameters ψ and ς (see (3.3) in Section 3.2.1.1 for the definition of ψ and ς) that minimize our objective function in (4.17). We shall denote the degree distribution obtained from this optimization process by $\Omega^{(1)}$.

In the second optimization we carry out we set no constraints at all on the degree distribution, except for the design constraints on the average and maximum output degree. We refer to the distribution obtained by this optimization method as $\Omega^{(2)}$.

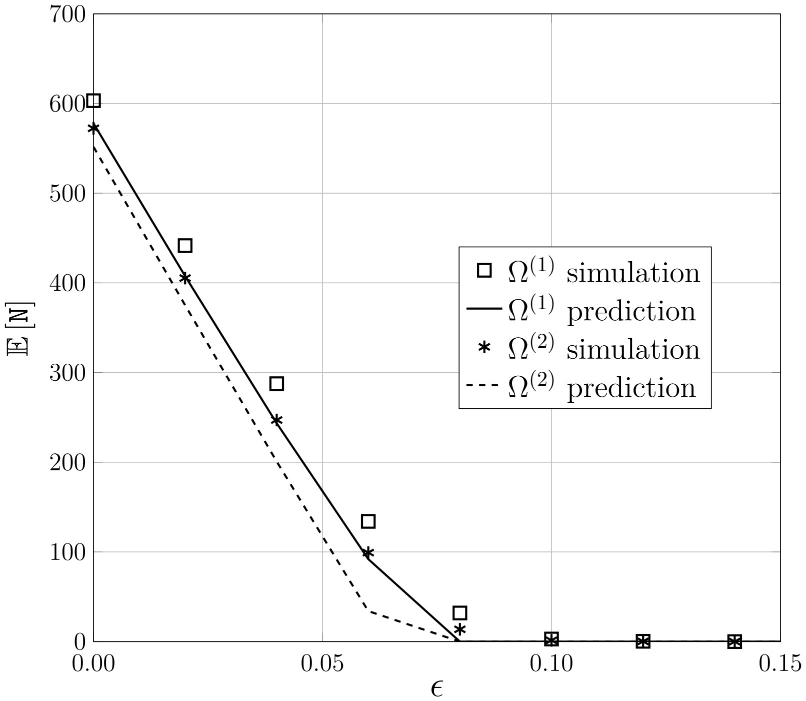


Fig. 4.18 Average number of inactivations needed for decoding vs. ϵ. The solid and dashed lines represent the predicted number of inactivations, $\mathbb{E}[\hat{\mathtt{N}}]$ for $\Omega^{(1)}$ and $\Omega^{(2)}$, respectively. The markers denote the average number of inactivations $\mathbb{E}[\mathtt{N}]$ obtained by Monte Carlo simulations.

Figure 4.18 shows the number of inactivations needed for decoding as a function of ϵ for $\Omega^{(2)}$ and $\Omega^{(1)}$, which has parameters $\varsigma = 0.05642$ and $\psi = 0.0317$. If we compare the number of inactivations needed by $\Omega^{(2)}$ and $\Omega^{(1)}$ we can observe how the $\Omega^{(2)}$, the result of the unconstrained optimization, requires slightly less inactivations. In the figure we can also observe how the estimation of the number of inactivations $\mathbb{E}\left[\hat{\mathtt{N}}\right]$ lies slightly below $\mathbb{E}\left[\mathtt{N}\right]$, which is an effect that had already been observed in Section 4.2.3.

For the sake of completeness, the probability of decoding failure for $\Omega^{(1)}$ and $\Omega^{(2)}$ is provided in Figure 4.19. It can be observed how $\underline{\mathsf{P}}_{\mathsf{F}}$ is below the target value $P_f^* = 10^{-2}$ in both cases, being the probability of decoding failure lower for the truncated RSD.

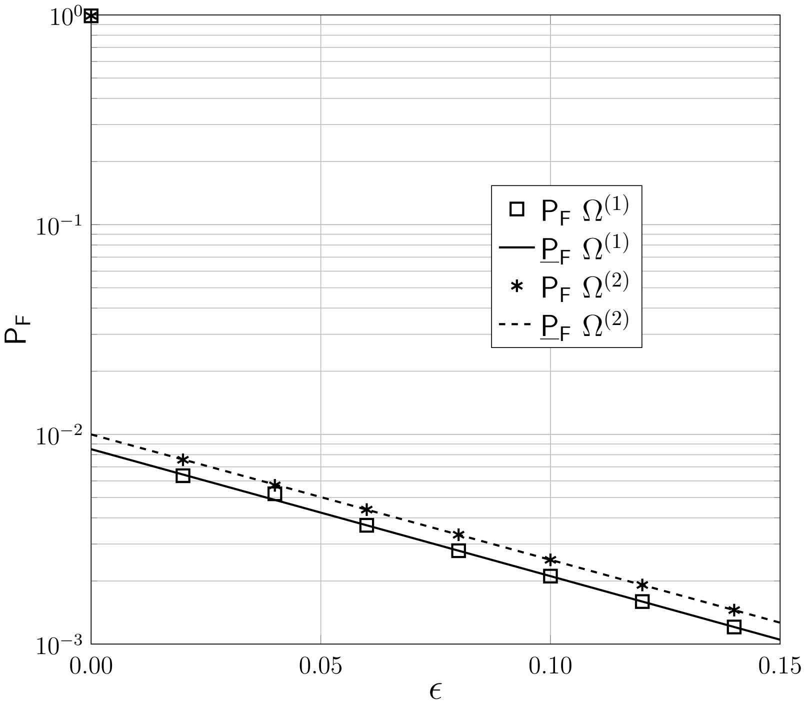


Fig. 4.19 Probability of decoding failure, $\mathsf{P_F}$ vs. ϵ for $\Omega^{(1)}$ and $\Omega^{(2)}$. The lines represent the lower bound $\underline{\mathsf{P}}_\mathsf{F}$ and markers denote simulation results.

4.4 Summary

In this chapter we have considered LT codes under inactivation decoding, an ML decoding algorithm belonging to the family of the structured or intelligent Gaussian elimination algorithms. Inactivation decoding aims at reducing the size of the system of equations that needs to be solved with standard Gaussian elimination (number of inactivations). The focus of the chapter is on the decoding complexity under inactivation decoding. In Section 4.2.1 we presented a first order analysis of LT codes under inactivation decoding that provides the expected number of inactivations given an output degree distribution. The analysis is based on a dynamic programming approach that models the decoder as a finite state machine. This model was extended in Section 4.2.2 in order to obtain the probability distribution of the number of inactivations. Section 4.2.3 presented an approximate low complexity method to estimate the expected number of inactivations. Finally in Section 4.3 we showed how these methods can be used to numerically design degree distributions to minimize the number of inactivations while fulfilling a set of design constraints.

Chapter 5

Raptor Codes under Inactivation Decoding

Within this chapter we consider Raptor codes under inactivation decoding. In Section 5.1 we develop upper bounds to the probability of decoding failure of q-ary Raptor codes under ML decoding using the weight enumerator of the outer code (precode), or its expected weight enumerator in case the outer code is not deterministic but drawn at random from an ensemble of codes. The bounds are shown to be tight, specially in the error floor region, by means of simulations. In Section 5.2 we consider the decoding complexity of Raptor codes under inactivation decoding, which is an ML decoding algorithm. More concretely, we provide a heuristic method to approximate the number of inactivations needed for decoding. In Section 5.3 we show how the results in this chapter can be used for Raptor code design by means of an example. Finally in Section 5.4 we summarize our contributions.

5.1 Performance under ML Decoding

The probability of decoding failure of Raptor codes under ML decoding[1] has been subject of study in several works. In [70] upper and lower bounds to the intermediate symbol erasure rate were derived for Raptor codes with outer codes in which the elements of the parity check matrix are independent and identically distributed (i.i.d.) Bernoulli random variables. This work was extended in [71] by deriving an approximation to the performance of Raptor codes under ML decoding under the assumption that the

[1] Inactivation decoding is a maximum likelihood decoding algorithm.

number of erasures correctable by the outer code is small. Thus, this approximation holds only when the rate of the outer code is sufficiently high. In [51] it was shown by means of simulations how the error probability of q-ary Raptor codes is very close to that of q-ary linear random fountain codes. In [72] upper and lower bounds to the probability of decoding failure of Raptor codes where derived. The outer codes considered in [72] are systematic binary linear random codes.

Although a number of works has studied the probability of decoding failure of Raptor codes, to the best of the knowledge of the author, the available results hold only for specific binary outer codes (see [70, 72, 38]). In this section we derive an upper bound to the probability of decoding failure of q-ary Raptor codes using the weight enumerator of the outer code (precode), or its expected weight enumerator in case the outer code is not deterministic but drawn at random from an ensemble of codes. Thus, the bounds derived in this chapter are generic and can be applied to any Raptor code, provided that the weight enumerator of the outer code is known.

In this section we consider ensembles of q-ary Raptor codes under ML decoding. More specifically, Raptor codes constructed over $\mathbb{F}_q$ with an (h, k) outer linear block code $\mathcal{C}$ are considered. Hence, the k input (or source) symbols, $\mathbf{u} = (u_1,\ u_2,\ \ldots, u_k)$, belong to $\mathbb{F}_q$.

Denoting by $\mathbf{G}_{\text{o}}$ the employed generator matrix of the outer code, of dimension $(k \times h)$ and with elements in $\mathbb{F}_q$, the intermediate symbols can be expressed as

$$\mathbf{v} = \mathbf{u}\mathbf{G}_{\text{o}}.$$

Note that, by definition, $\mathbf{v} = (v_1,\ v_2,\ \ldots, v_h) \in \mathcal{C}$. The intermediate symbols serve as input to a q-ary LT encoder, that generates an unlimited number of output symbols, $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, where n can grow unbounded. Again, the elements of $\mathbf{c}$ belong to $\mathbb{F}_q$. For any n the output symbols can be expressed as

$$\mathbf{c} = \mathbf{v}\mathbf{G}_{\text{LT}} = \mathbf{u}\mathbf{G}_{\text{o}}\mathbf{G}_{\text{LT}}$$

where $\mathbf{G}_{\text{LT}}$ is an $(h \times n)$ with elements in $\mathbb{F}_q$. Each column of $\mathbf{G}_{\text{LT}}$ is associated with c_i. More specifically, each column of $\mathbf{G}_{\text{LT}}$ is generated by first selecting an output degree d according to the degree distribution Ω, and then selecting d different indices uniformly at random between 1 and h. Finally, the elements of the column corresponding to these indices are drawn independently and uniformly at random from $\mathbb{F}_q \backslash \{0\}$, while all other elements of the column are set to zero.

We consider the transmission over a QEC at the output of which each transmitted symbol is either correctly received or erased. Denoting by m the number of output symbols collected by the receiver of interest, and expressing it as $m = k + \delta$, where δ is the absolute receiver overhead. Let us denote by $\mathbf{y} = (y_1, y_2, \dots, y_m)$ the m received output symbols. Let us denote by $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ the set of indices corresponding to the m non-erased symbols, we have

$$y_j = c_{i_j}.$$

In this section we will assume that ML Raptor decoding is performed by solving the system of equations[2]

$$\mathbf{y} = \mathbf{v}\mathbf{G}_{\mathrm{R}}$$

where

$$\mathbf{G}_{\mathrm{R}} = \mathbf{G}_{\mathrm{o}}\tilde{\mathbf{G}}_{LT} \tag{5.1}$$

with $\tilde{\mathbf{G}}_{LT}$ given by the m columns of $\mathbf{G}_{\mathrm{LT}}$ with indices in $\mathcal{I}$.

5.1.1 Upper Bounds on the Error Probability

An upper bound on the probability of failure P_{F} of a Raptor code constructed over $\mathbb{F}_q$ as a function of the receiver overhead δ is established in the next Theorem.

Theorem 1. *Consider a Raptor code constructed over $\mathbb{F}_q$ with an (h, k) outer code $\mathcal{C}$ characterized by a weight enumerator A°, and an inner LT code with output degree distribution Ω. The probability of decoding failure under optimum (ML) erasure decoding given that $m = k + \delta$ output symbols have been collected by the receiver can be upper bounded as*

$$\mathsf{P}_{\mathsf{F}} \leq \sum_{l=1}^{h} A^{\circ}_{l} \pi_l^{k+\delta}$$

[2] In practice Raptor decoding is performed by solving the system of equations in (3.9) that involves the constraint matrix. The two systems of equations are equivalent. In this section we take the system of equations involving the generator matrix of the Raptor code for convenience.

where π_l is the probability that a generic output symbol Y is equal to 0 given that the vector $\mathbf{v}$ of intermediate symbols has Hamming weight l. The expression of π_l is

$$\pi_l = \frac{1}{q} + \frac{q-1}{q} \sum_{j=1}^{d_{\max}} \Omega_j \frac{\mathcal{K}_j(l;h,q)}{\mathcal{K}_j(0;h,q)} \tag{5.2}$$

where $\mathcal{K}_j(l;h,q)$ is the Krawtchouk polynomial of degree j with parameters h and q, defined as [73]

$$\mathcal{K}_k(x;n,q) = \sum_{j=0}^{k} (-1)^j \binom{x}{j} \binom{n-x}{k-j} (q-1)^{k-j}.$$

Proof. An optimum (ML) decoder solves the linear system of equations in (5.1). Decoding will fail whenever the system does not admit a unique solution, that is, if and only if $\mathsf{rank}(\mathbf{G}_{\mathrm{R}}) < k$, i.e. if $\exists\, \mathbf{u} \in \mathbb{F}_q^k \backslash \{\mathbf{0}\}$ s.t. $\mathbf{u}\mathbf{G}_{\mathrm{R}} = \mathbf{0}$. Let us consider two vectors $\mathbf{u} \in \mathbb{F}_q^k, \mathbf{v} \in \mathbb{F}_q^h$. Define $E_{\mathbf{u}}$ as the event $\mathbf{u}\mathbf{G}_{\mathrm{o}}\tilde{\mathbf{G}}_{LT} = \mathbf{0}$. Similarly, Define $E_{\mathbf{v}}$ as the event $\mathbf{v}\tilde{\mathbf{G}}_{LT} = \mathbf{0}$. We have

$$\mathsf{P_F} = \Pr\left\{ \bigcup_{\mathbf{u} \in \mathbb{F}_q^k \backslash \{\mathbf{0}\}} E_{\mathbf{u}} \right\} = \Pr\left\{ \bigcup_{\mathbf{v} \in \mathcal{C} \backslash \{\mathbf{0}\}} E_{\mathbf{v}} \right\} \tag{5.3}$$

where we made use of the fact that due to linearity, the all zero intermediate word is only generated by the all zero input vector.

If we develop (5.3), we have

$$\mathsf{P_F} = \Pr\left\{ \bigcup_{l=1}^{h} \bigcup_{\mathbf{v} \in \mathcal{C}_l} E_{\mathbf{v}} \right\} \tag{5.4}$$

where, $\mathcal{C}_l$ is the set of codewords in $\mathcal{C}$ of Hamming weight l, formally,

$$\mathcal{C}_l = \{\mathbf{v} \in \mathcal{C} : w_H(\mathbf{v}) = l\}\,.$$

Let L be a discrete random variable representing the Hamming weight of vector $\mathbf{v} \in \mathcal{C}$. Moreover, let J and I be discrete random variables representing the degree of the generic output symbol y, and the number of non-zero neighbors of such intermediate symbol, respectively. Note that $I \leq L$. By making use of Boole's inequality (also known as the union bound) it is possible to upper bound (5.4) as

$$\begin{aligned}
\mathsf{P_F} &\leq \sum_{l=1}^{h} \Pr\left\{ \bigcup_{\mathbf{v} \in \mathcal{C}_l} E_{\mathbf{v}} \right\} \\
&\leq \sum_{l=1}^{h} A_l^{\circ} \Pr\left\{ E_{\mathbf{v}} | L = l \right\} \ .
\end{aligned} \tag{5.5}$$

Since output symbols are independent of each other, we have

$$\Pr\left\{ E_{\mathbf{v}} | L = l \right\} = \pi_l^{k+\delta}$$

where $\pi_l = \Pr\{y = 0 | L = l\}$ is the conditional probability that the generic output symbol y is equal to $0 \in \mathbb{F}_q$ given that $\mathbf{v} \in \mathcal{C}_l$. An expression for π_l may be obtained observing that

$$\begin{aligned}
\pi_l &= \sum_{j=1}^{d_{\max}} \Pr\{y = 0 | L = l, J = j\} \Pr\{J = j | L = l\} \\
&\overset{\text{(a)}}{=} \sum_{j=1}^{d_{\max}} \Omega_j \Pr\{y = 0 | L = l, J = j\} \\
&\overset{\text{(b)}}{=} \sum_{j=1}^{d_{\max}} \Omega_j \sum_{i=0}^{\min\{j,l\}} \Pr\{y = 0 | I = i\} \Pr\{I = i | L = l, J = j\}
\end{aligned}$$

where equality (a) is due to $\Pr\{J = j | L = l\} = \Pr\{J = j\} = \Omega_j$ and equality (b) to $\Pr\{y = 0 | L = l, J = j, I = i\} = \Pr\{y = 0 | I = i\}$. Letting $\vartheta_{i,l,j} = \Pr\{I = i | L = l, J = j\}$, since the j intermediate symbols are chosen uniformly at random by the LT encoder we have

$$\vartheta_{i,l,j} = \frac{\binom{l}{i}\binom{h-l}{j-i}}{\binom{h}{j}} \ . \tag{5.6}$$

Denoting $\Pr\{y = 0|I = i\}$ by φ_i and observing that, due to the elements of $\mathbf{G}_{\mathrm{R}}$ being i.i.d. and uniformly drawn in $\mathbb{F}_q \setminus \{0\}$, on invoking Lemma 3 in Appendix B[3] we have

$$\varphi_i = \frac{1}{q}\left(1 + \frac{(-1)^i}{(q-1)^{i-1}}\right). \tag{5.7}$$

Hence, π_l is given by

$$\pi_l = \sum_{j=1}^{d_{\max}} \Omega_j \sum_{i=0}^{\min\{j,l\}} \vartheta_{i,l,j}\, \varphi_i$$

where $\vartheta_{i,l,j}$ and φ_i are given by (5.6) and (5.7), respectively.

Expanding this expression and rewriting it using Krawtchouk polynomials and making use of the Chu-Vandermonde identity[4], one obtains (5.2). □

The following theorem makes the bound in Theorem 1 tighter for $q > 2$. For $q = 2$ the following theorem is equivalent to Theorem 1 .

Theorem 2. *Consider a Raptor code constructed over $\mathbb{F}_q$ with an (h, k) outer code $\mathcal{C}$ characterized by a weight enumerator A°, and an inner LT with output degree distribution Ω. The probability of decoding failure under optimum erasure decoding given that $m = k + \delta$ output symbols have been collected by the receiver can be upper bounded as*

$$\mathsf{P_F} \leq \sum_{l=1}^{h} \frac{A^{\circ}_l}{q-1} \pi_l^{k+\delta}$$

Proof. The bound (5.5) can be tightened by a factor $q - 1$ exploiting the fact that for a linear block code $\mathcal{C}$ constructed over $\mathbb{F}_q$, if $\mathbf{c}$ is a codeword, $\alpha\mathbf{c}$ is also a codeword, $\forall \alpha \in \mathbb{F}_q \backslash \{0\}$, [47]. □

Remark 1. *The upper bound in Theorem 2 can also be applied to LT codes. In that case, A°_l needs to be replaced by the total number of sequences of Hamming weight l*

[3] Lemma 3 is only valid for $q = 2^m$, the case of most interest for practical purposes. The proof of the general case is a trivial extension of Lemma 3. The result in Lemma 3 can also be found in [56], where the proof was derived using a different technique from the one used in Appendix B (see Appendix B for more details).

[4] The Chu-Vandermonde identity, or Vandermonde's convolution formula is given by

$$\binom{x+a}{n} = \sum_{k=0}^{n} \binom{x}{k}\binom{a}{n-k}$$

and length k,

$$A_l^{\mathsf{o}} = \binom{k}{l}(q-1)^{l-1}.$$

The upper bound obtained for LT codes coincides with the bound in [56] (Theorem 1), where only LT codes are considered. Thus, we can regard Theorem 2 as an extension of the work in [56] to Raptor codes.

5.1.2 Random Outer Codes from Linear Parity-Check Based Ensembles

Both Theorem 1 and Theorem 2 apply to the case of a deterministic outer code. In this section we extend these results to the case of a random outer code drawn from an ensemble of codes. Specifically, a parity-check based ensemble of outer codes is considered, denoted by $\mathscr{C}^{\mathrm{o}}$, defined by a random matrix of size $(h-k)\times h$ whose elements belong to $\mathbb{F}_q$. A linear block code of length h belongs to $\mathscr{C}^{\mathrm{o}}$ if and only if at least one of the instances of the random matrix is a valid parity-check matrix for it. Moreover, the probability measure of each code in the ensemble is the sum of the probabilities of all instances of the random matrix which are valid parity-check matrices for that code. Note that all codes in $\mathscr{C}^{\mathrm{o}}$ are linear, have length h, and have dimension $k' \geq k$[5].

In the following the expression "Raptor code ensemble" is used to refer to the set of Raptor codes obtained by concatenating an outer code belonging to the parity-check based ensemble $\mathscr{C}^{\mathrm{o}}$ with an LT encoder having distribution Ω. This ensemble shall be denoted as $(\mathscr{C}^{\mathrm{o}}, \Omega)$. The following theorem extends the result in Theorem 2 to ensembles of Raptor codes.

Theorem 3. *Consider a Raptor code ensemble $(\mathscr{C}^o, \Omega)$ and let $\mathsf{A} = \{\mathsf{A}_0, \mathsf{A}_1, \dots, \mathsf{A}_h\}$ be the expected weight enumerator of a code $\mathcal{C}$ that is randomly drawn from ensemble $\mathscr{C}^o$, i.e., let $\mathsf{A}_l = \mathbb{E}_{\mathcal{C}}[A_l(\mathcal{C})]$ for all $l \in \{0, 1, \dots, h\}$. Let $\bar{\mathsf{P}}_{\mathsf{F}}$ be the average probability of decoding failure of the Raptor code ensemble obtained by concatenating a code $\mathcal{C}$ randomly drawn from ensemble $\mathscr{C}^o$ with the LT encoder with degree distribution Ω, under maximum likelihood (ML) erasure decoding and given that $m = k + \delta$ output symbols have been collected by the receiver. Then*

$$\bar{\mathsf{P}}_{\mathsf{F}} \leq \sum_{l=1}^{h} \frac{\mathsf{A}_l}{q-1} \pi_l^{k+\delta}.$$

[5] This comes from the fact that the parity-check matrix can be rank deficient.

Proof. We can express the average probability of decoding failure as

$$\bar{\mathsf{P}}_{\mathsf{F}} = \mathbb{E}_{\mathscr{C}^{\mathrm{o}}}[\mathsf{P}_{\mathsf{F}}(\mathcal{C})]$$

where $\mathsf{P}_{\mathsf{F}}(\mathcal{C})$ is the probability of decoding failure when outer code $\mathcal{C}$ is selected as outer code, and the expectation is taken over all the outer codes $\mathcal{C}$ in the ensemble $\mathscr{C}^{\mathrm{o}}$.

From Theorem 2 we have

$$\bar{\mathsf{P}}_{\mathsf{F}} \leq \mathbb{E}\left[\sum_{l=1}^{h} \frac{A^{\mathrm{o}}_{l}(\mathcal{C})}{q-1} \pi_l^{k_{\mathcal{C}}+\delta}\right] \tag{5.8}$$

where $A^{\mathrm{o}}_{l}(\mathcal{C})$ is the number of codewords of weight l in $\mathcal{C}$ and $k_{\mathcal{C}}$ is the dimension of $\mathcal{C}$.

For all codes $\mathcal{C}$ in the ensemble $\mathscr{C}^{\mathrm{o}}$ we have $k_{\mathcal{C}} \geq k$. Furthermore, since π_l is a probability we have $\pi_l \leq 1$ and we can write

$$\pi_l^{k_{\mathcal{C}}+\delta} \leq \pi_l^{k+\delta}$$

which allows us to upper bound (5.8) as

$$\bar{\mathsf{P}}_{\mathsf{F}} \leq \mathbb{E}\left[\sum_{l=1}^{h} \frac{A^{\mathrm{o}}_{l}(\mathcal{C})}{q-1} \pi_l^{k+\delta}\right]$$

which by linearity of the expectation becomes

$$\bar{\mathsf{P}}_{\mathsf{F}} \leq \sum_{l=1}^{h} \frac{\mathbb{E}\left[A^{\mathrm{o}}_{l}(\mathcal{C})\right]}{q-1} \pi_l^{k+\delta} = \sum_{l=1}^{h} \frac{\mathsf{A}_l}{q-1} \pi_l^{k+\delta}$$

□

Theorem 1 can be extended in the same way as Theorem 2 to consider the case when the outer code is drawn from an ensemble of codes.

5.1.3 Numerical Results

All the results presented in this section use the LT output degree distribution from standard R10 Raptor codes, [30, 31] with degree generator polynomial:

$$\begin{aligned}\Omega(\mathrm{x}) = {} & 0.0098\mathrm{x} + 0.4590\mathrm{x}^2 + 0.2110\mathrm{x}^3 + 0.1134\mathrm{x}^4 \\ & + 0.1113\mathrm{x}^{10} + 0.0799\mathrm{x}^{11} + 0.0156\mathrm{x}^{40}.\end{aligned} \tag{5.9}$$

The bound used in the different figures in this section is that given in Theorem 2 for deterministic outer codes. When considering outer codes drawn for an ensemble the bound in Theorem 3 is used.

5.1.3.1 Hamming outer code

In this section we consider Raptor codes with binary Hamming outer codes. The weight enumerator of a Hamming code can be derived easily using the following recursion,

$$(i+1)\,A_{i+1} + A_i + (n-i+1)\,A_{i-1} = \binom{n}{i}$$

with $A_0 = 1$ and $A_1 = 0$ [73]. The expression obtained from this recursion can then be used together with Theorem 1 to derive an upper bound on the probability of decoding failure.

In Figure 5.1 we show the probability of decoding failure for a Raptor code with a $(63, 57)$ binary Hamming outer code as a function of the absolute overhead, δ. In order to obtain the average probability of failure, Monte Carlo simulations were run until 200 errors were collected. It can be observed how the upper bound is tight, specially in the error floor region (when δ is large).

5.1.3.2 Linear random outer code

In this section an ensemble of Raptor codes is consider where the outer code is selected from the q-ary linear random ensemble. The average weight enumerator of the linear random ensemble was first derived in [16] for the binary case and then in [74] for the q-ary case and has the expression

$$\mathsf{A}_l = \binom{h}{l} q^{-h(1-\mathsf{r_o})}(q-1)^l.$$

In order to simulate the average probability of decoding failure of the ensemble, 6000 different outer codes were generated. For each outer code and overhead value 10^3 decoding attempts were carried out. The average probability of decoding failure was obtained averaging the probabilities of decoding failure obtained with the different outer codes. Note that the objective of the simulation was not characterizing the performance of every individual code but to characterize the average performance of the ensemble. In order to select the outer code an $(h-k) \times h$ parity check matrix

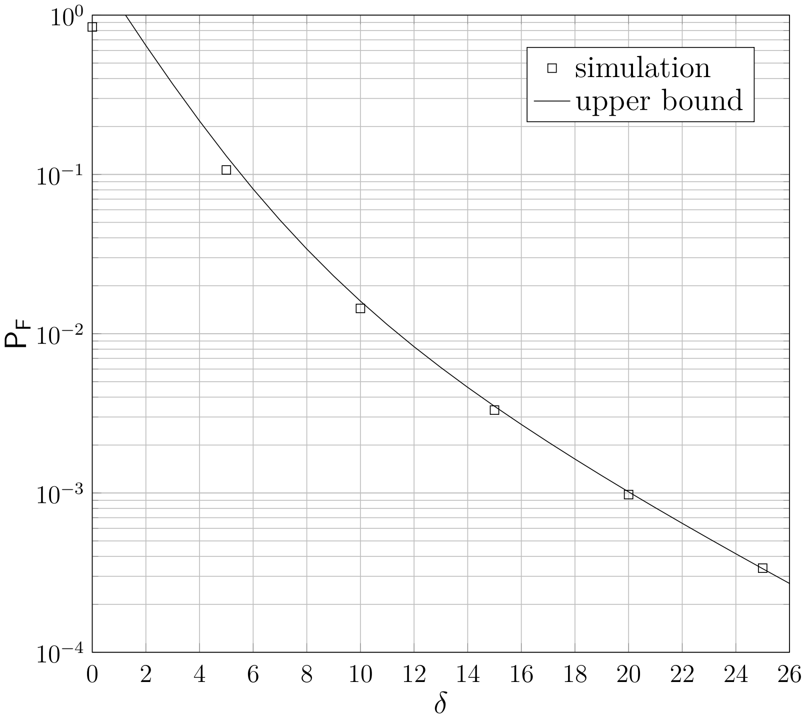


Fig. 5.1 Probability of decoding failure $\mathsf{P_F}$ vs. absolute overhead for a Raptor code with a $(63, 57)$ Hamming outer code and LT output degree distribution given in (5.9). The solid line denotes the upper bound on the probability of decoding failure in Theorem 2. The markers denote simulation results.

was selected at random by generating each of its elements according to a uniform distribution in $\mathbb{F}_q$.

Figure 5.2 shows the simulation results for Raptor codes with a linear random outer code with $k = 64$ input symbols and $h = 70$ intermediate symbols and the degree distribution in (5.9). Two different ensembles were considered, a binary one and one constructed over $\mathbb{F}_4$. We can observe how in both cases the bounds hold and are tight except for very small values of δ.

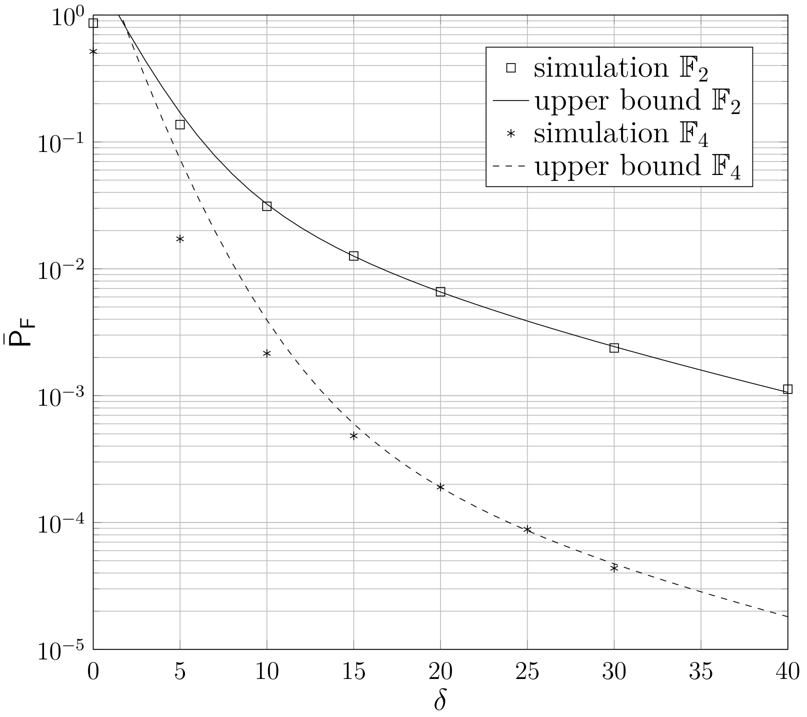


Fig. 5.2 Expected probability of decoding failure $\bar{\mathsf{P}}_{\mathsf{F}}$ vs. absolute overhead for two Raptor code ensembles where the outer code is selected from the (70,64) linear random ensemble constructed over $\mathbb{F}_2$ and $\mathbb{F}_4$ and with LT output degree distribution given in (5.9). The solid and dashed lines denote the upper bound on the probability of decoding failure for the ensembles constructed over $\mathbb{F}_2$ and $\mathbb{F}_4$ respectively (see Theorem 3). The square and asterisk markers denote simulation results for $\mathbb{F}_2$ and $\mathbb{F}_4$ respectively.

5.2 Inactivation Decoding Analysis

In this section we will consider Raptor codes under inactivation decoding, the efficient ML decoding algorithm that is described in Section 4.1 within the context of LT codes. For simplicity, in this section we will consider only binary Raptor codes, being the extension to non-binary Raptor codes straightforward.

Let us recall, that ML decoding of LT codes consists of solving the system of equations in (3.4):

$$\mathbf{y}^T = \tilde{\mathbf{G}}^T \mathbf{v}^T$$

where $\mathbf{y}$ is the (row) vector of received output symbols, $\mathbf{v}$ is the (row) vector of source symbols and $\tilde{\mathbf{G}}^T$ is the transposed generator matrix of the LT code after removing the rows associated to output symbols that were erased by the channel. The matrix

$\tilde{\mathbf{G}}^T$ has dimensions $m \times k$, m being the number of output symbols collected and k the number of source symbols.

ML decoding of Raptor codes consists of solving the system of equations in (3.9). That is,

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{y^T} \end{bmatrix} = \mathbf{M}\, \mathbf{v}^T$$

where $\mathbf{v}$ is a row vector representing the intermediate symbols, that are the input to the LT encoder, $\mathbf{y}$ is a column vector representing the received output symbols and $\mathbf{M}$ is the constraint matrix of the Raptor code with dimension $((h-k+m) \times h)$. Vector $\mathbf{z}$ is a $(h-k) \times 1$ column zero vector (see Section 3.3.1 for more details).

If we compare the two systems of equations, we can see how Raptor ML decoding is very similar to ML decoding of an LT code with h source symbols and $h-k+m$ output symbols. The role of matrix $\tilde{\mathbf{G}}$ for LT codes is played by the constraint matrix of the Raptor code, $\mathbf{M}$. The main difference is that while for LT codes all the rows of $\tilde{\mathbf{G}}$ are independent and identically distributed according to the degree distribution Ω, this is no longer true for matrix $\mathbf{M}$ (for a generic Raptor code). Let us recall that the constraint matrix of a Raptor code is defined as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{H}_{\text{o}} \\ \tilde{\mathbf{G}}_{LT}^T \end{bmatrix}$$

where $\mathbf{H}_{\text{o}}$ is the parity check matrix of the outer code, and $\tilde{\mathbf{G}}_{LT}$ is a binary matrix that corresponds to the generator matrix of the LT code after removing the columns associated with the output symbols that were erased by the channel. Thus, while the rows in $\tilde{\mathbf{G}}_{LT}$ are independent and identically distributed this is not true for matrix $\mathbf{M}$.

Let us introduce the following definition.

Definition 12 (Surrogate LT code)**.** *Consider a Raptor code with outer code parity-check matrix $\mathbf{H}_o$ and an inner LT code with degree distribution Ω and assume the receiver has collected $m = k + \delta$ output symbols. The surrogate LT code of the Raptor code is defined as an LT code with h input symbols, m output symbols, and degree distribution $\hat{\Omega}$ given by the expected Hamming weight distribution of the rows of the constraint matrix M of the Raptor code. Formally*

$$\hat{\Omega} = \frac{h-k}{h-k+m}\Theta + \frac{m}{h-k+m}\Omega = \hat{\Omega} = \frac{h-k}{h+\delta}\Theta + \frac{m}{h+\delta}\Omega$$

where $\Theta = \{\Theta_1, \Theta_2, \dots, \Theta_h\}$*, being* Θ_i *the fraction of rows of Hamming weight* i *in* $\mathbf{H}_o$*.*

Remark 2. *The degree distribution of the surrogate LT code,* $\hat{\Omega}$ *depends on the receiver overhead* δ*.*

Given the similarity between inactivation decoding of Raptor and LT codes, it is possible to approximate inactivation decoding of a Raptor code as inactivation decoding of its surrogate LT code. Using this heuristic approximation we will show how the approaches derived in Sections 4.2.1, 4.2.2 and 4.2.3 for LT codes can be adapted to approximate inactivation decoding of Raptor codes with a reasonable accuracy.

5.2.1 Raptor Codes with Linear Random Outer Codes

In the case of Raptor codes with a linear Random outer code the constraint matrix corresponds to:

$$\mathbf{M} = \begin{bmatrix} \mathbf{H}_{\mathrm{o}} \\ \tilde{\mathbf{G}}_{LT}^{T} \end{bmatrix}$$

where

- $\mathbf{H}_{\mathrm{o}}$ is the parity check matrix of a linear random code with size $((h-k) \times h)$. The Hamming weight of each row corresponds to a binomial random variable with parameters h and $1/2$.
- $\tilde{\mathbf{G}}_{LT}$ is a $(h \times m)$ binary matrix which defines the relation between the intermediate symbols and the output symbols due to the LT encoding. The Hamming weight of each column corresponds to the output degree distribution of the inner LT code Ω.

Thus the Hamming weight distribution of $\mathbf{H}_{\mathrm{o}}$ corresponds to

$$\Theta_i = \mathcal{B}(h, 1/2)$$

where $\mathcal{B}(h, 1/2)$ is a binomial distribution with parameters h and $1/2$. Therefore, the degree distribution of the surrogate LT code corresponds to

$$\hat{\Omega} = \frac{m}{h-k+m}\Omega + \frac{h-k}{h-k+m}\mathcal{B}(h, 1/2)$$

For illustration, in Figure 5.3 we provide an example of degree distribution of the surrogate LT code for a Raptor code with a $(106, 80)$ linear random outer code with

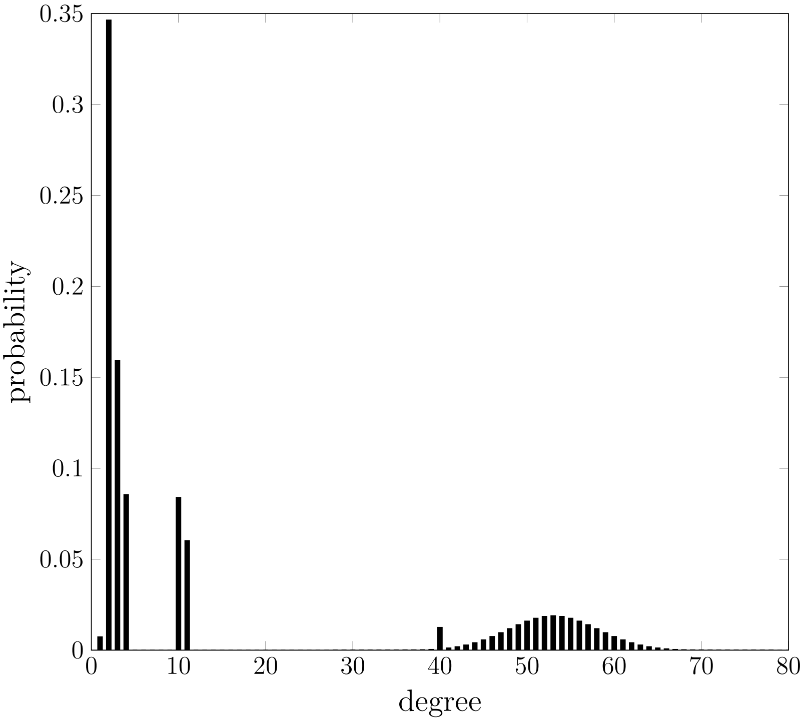


Fig. 5.3 Surrogate LT degree distribution $\hat{\Omega}$ for a Raptor code with a $(106, 80)$ linear random outer code with $m = 80$ and degree distribution Ω^{R10}.

$m = 80$ and degree distribution Ω^{R10}. The contribution of the outer code can be clearly distinguished, it corresponds to the bell shaped curve around degree $d = 53$.

Figure 5.4 shows the average number of inactivations vs. the absolute receiver overhead δ for a Raptor code with a $(233, 200)$ linear random precode with degree distribution Ω^{R10}. The figure shows results obtained by Monte Carlo simulations and the approximations obtained using the methods in Sections 4.2.1 and 4.2.3 for the surrogate LT code. The match between the simulation results and the approximation is good.

Finally, Figure 5.5 shows the distribution of the number of inactivations for a Raptor code with a $(233, 200)$ linear random outer code with $m = 200$ and degree distribution Ω^{R10} and the approximation obtained using the method in Section 4.2.2 for the surrogate LT code. We can observe how the estimation of the distribution of the number of inactivations is not very accurate. While the average value is estimated correctly, the actual distribution of the number of inactivations is more concentrated around the mean than its estimation. A possible explanation for this effect is that the Raptor code has a constant number large Hamming weight rows in its constraint

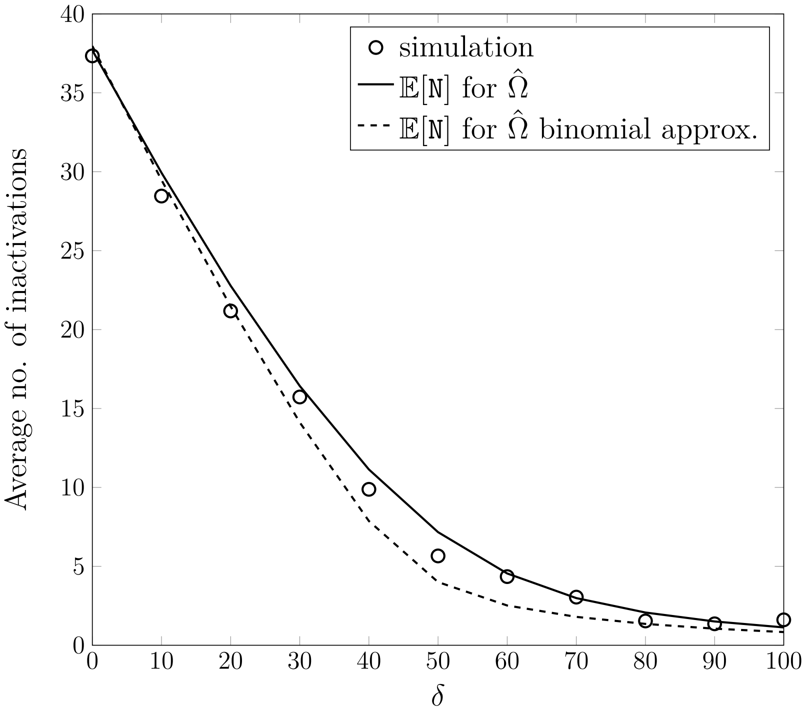


Fig. 5.4 Average number of inactivations vs. absolute receiver overhead δ for a Raptor code with a $(233, 200)$ linear random outer code with degree distribution Ω^{R10}. The markers represent the results of Monte Carlo simulations. The solid and dashed lines represent the number of inactivations for the surrogate LT code using the methods Sections 4.2.1 and 4.2.3 respectively.

matrix, which correspond to the parity check matrix of the outer code. However, its surrogate LT code implicitly makes the assumption that the number of large weight rows is random. Thus, it also considers realizations with too many/few large Hamming weight rows (output symbols), leading to a higher dispersion (less concentration) of the number of inactivations around the mean value.

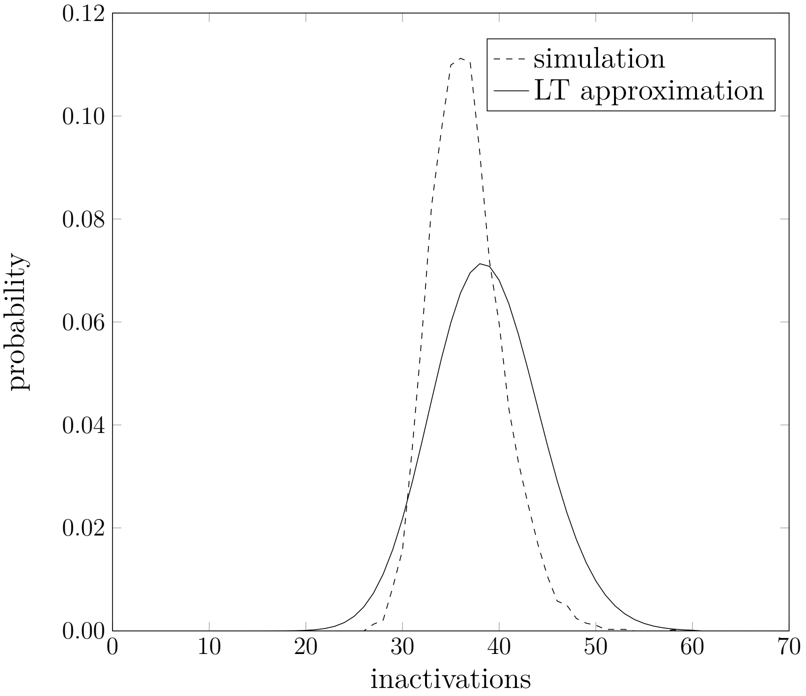


Fig. 5.5 Distribution of the number of inactivations for a Raptor code with a $(233, 200)$ linear random outer code with $m = 200$ and degree distribution Ω^{R10}. The dashed line represents the results of Monte Carlo simulations. The solid line represents the estimated number of inactivations using the method in Section 4.2.2 for the surrogate LT code.

5.2.2 R10 Raptor Codes

In the case of R10 Raptor codes, the precode is a concatenation of two systematic codes, an LDPC code and an HDPC code (see Section 3.3.2). Hence, two different parts can be distinguished in the parity check matrix of the precode:

- LDPC part. There are s_{R10} rows associated to LDPC redundant symbols. The Hamming weight of each row is approximately $3\lfloor k/s_{\mathrm{R10}}\rceil + 1$, where $\lfloor x\rceil$ denotes the closest integer to x.

- HDPC part. There are h_{R10} rows associated to HDPC redundant symbols. The Hamming weight of each row is approximately $\lfloor (k + s_{\mathrm{R10}})/2\rceil + 1$

Thus, the distribution of the Hamming weight of the rows of $\mathbf{H}_{\mathrm{o}}$ is approximately

$$\Theta_i \approx \frac{s_{\mathrm{R10}}}{s_{\mathrm{R10}} + h_{\mathrm{R10}}}\mathcal{D}\left(3\left\lfloor\frac{k}{s_{\mathrm{R10}}}\right\rceil + 1\right) + \frac{h_{\mathrm{R10}}}{s_{\mathrm{R10}} + h_{\mathrm{R10}}}\mathcal{D}\left(\left\lfloor\frac{k + s_{\mathrm{R10}}}{2}\right\rceil + 1\right),$$

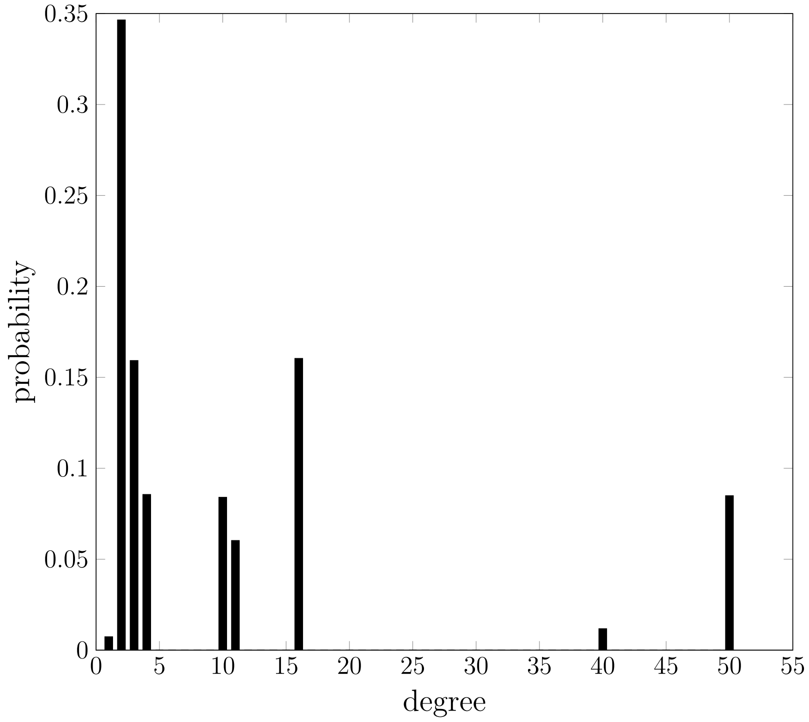


Fig. 5.6 Surrogate LT degree distribution $\hat{\Omega}$ for an R10 Raptor code with $k = 80$.

where $\mathcal{D}(i)$ denotes a (discrete) Kronecker delta at i. Therefore, the degree distribution of the surrogate LT code is approximately

$$\hat{\Omega} \approx \frac{m}{s_{\mathrm{R10}} + h_{\mathrm{R10}} - k + m} \Omega + \frac{s_{\mathrm{R10}}}{s_{\mathrm{R10}} + h_{\mathrm{R10}} + m} \mathcal{D}\left(3 \left\lceil \frac{k}{s_{\mathrm{R10}}} \right\rceil + 1\right) \\ + \frac{h_{\mathrm{R10}}}{s_{\mathrm{R10}} + h_{\mathrm{R10}} + m} \mathcal{D}\left(\left\lceil \frac{k + s_{\mathrm{R10}}}{2} \right\rceil + 1\right).$$

For illustration, in Figure 5.6 we provide the surrogate LT code degree distribution for a R10 Raptor code with $k = 80$ and $m = 80$. In this case the redundant LDPC symbols are modeled as degree 16 output symbols and the HDPC symbols by degree 50 output symbols.

Using the surrogate LT code approximation, the methods presented in Sections 4.2.1, 4.2.2 and 4.2.3 can be used to estimate the number of inactivations needed to complete Raptor decoding.

Figure 5.7 shows the average number of inactivations vs. the absolute receiver overhead δ for a R10 Raptor code with $k = 200$. The figure shows results obtained by Monte Carlo simulations and the approximations obtained using the methods in

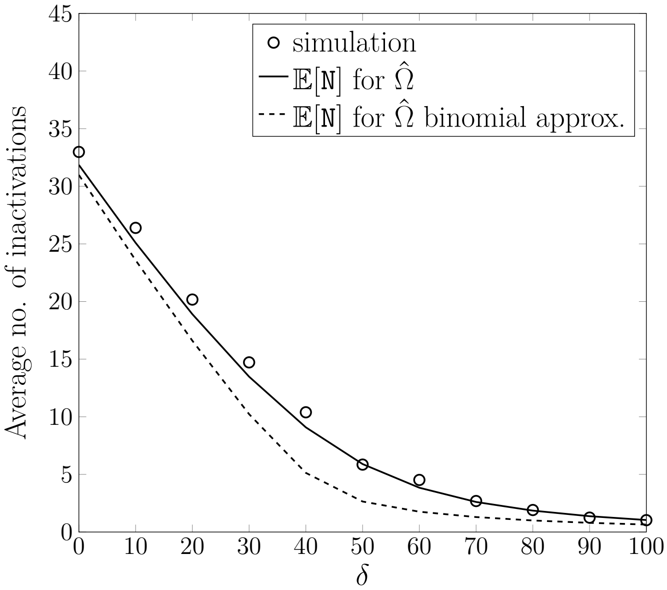


Fig. 5.7 Average number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code with $k = 200$. The markers represent the results of Monte Carlo simulations. The solid and dashed lines represent the estimated number of inactivations for the surrogate LT code using the methods in Sections 4.2.1 and 4.2.3 respectively.

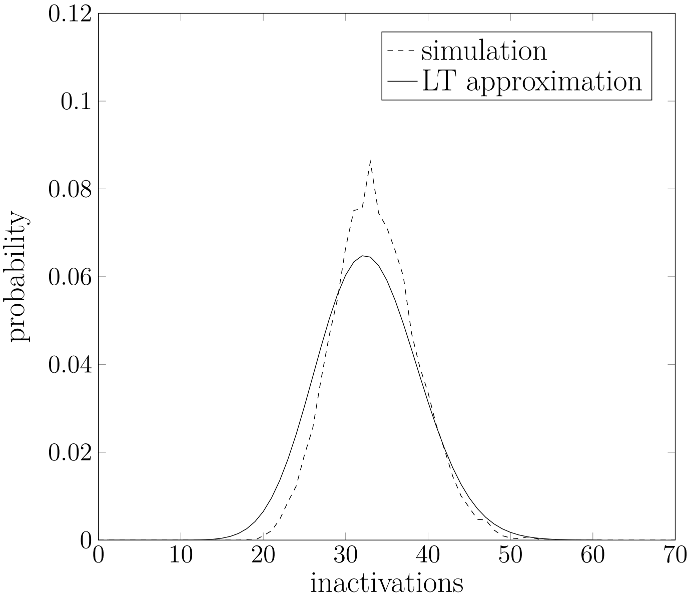


Fig. 5.8 Distribution of the number of inactivations for a R10 Raptor code with $k = 200$ and $m = 200$. The dashed line represents the results of Monte Carlo simulations. The solid line represents the estimated number of inactivations using the surrogate LT code approximation and the method in Section 4.2.2.

Sections 4.2.1 and 4.2.3 for the surrogate LT code. The match between the simulation results and the method in Section 4.2.3 is good.

Finally, Figure 5.8 shows the distribution of the number of inactivations for a R10 Raptor code with a $k = 200$ and $m = 200$ and its estimation using the surrogate LT code approximation and the method in Section 4.2.2. If we compare this figure with Figure 5.5 we can see how the estimation of the distribution of the number of inactivations works better for R10 Raptor codes than for codes with a linear random outer code.

5.2.3 Discussion

In this section we have proposed an approximate analysis of Raptor codes by introducing the concept of the surrogate LT code of a Raptor code. The simulation results presented in Sections 5.2.1 and 5.2.2 show how the approximation is reasonably good in order to estimate the expected number of inactivations. However, the approximation is not accurate enough to estimate the distribution of the number of inactivations. The reason for this deviation is the fact that the surrogate LT approximation makes implicitly the assumption that the rows of the constraint matrix $\mathbf{M}$ are independent and identically distributed. More concretely, it assumes there is a random number of large Hamming weight rows, which correspond to the rows in the parity check matrix of the outer code. However, for a Raptor code, the number of rows of $\mathbf{M}$ that correspond to the parity check matrix of the outer code is fixed.

5.3 Code Design

Within this section we provide a Raptor code design example. More concretely we design a degree distribution for the LT component of a binary Raptor code with a $(63, 57)$ outer Hamming code. The design goal is achieving a target probability of decoding failure lower than $\mathsf{P_F}^* = 10^{-3}$ for $\delta = 15$ while minimizing the number of inactivations needed for decoding at $\delta = 15$. Moreover, we will constraint the output degree distribution to have exactly the same maximum and average output degree as standard R10 Raptor codes ($\bar{\Omega} = 4.6314$ and $d_{\max} = 40$). Note that a constraint on the average output degree is equivalent to a constraint on the average encoding complexity /cost. Moreover, the constraint on the maximum output degree gives us control on the worst case encoding complexity. Furthermore we will constraint the output degree distribution to have the same support as the degree distribution of R10 raptor codes, that is, only degrees $1, 2, 3, 4, 10, 11$ and 40 will be assigned a probability larger than 0. These constraints are chosen to illustrate the fact that arbitrary constraints can be introduced in the code design.

The design of the LT output degree distribution is formulated as a numerical optimization problem. More concretely, the numerical optimization algorithm that is used is simulated annealing (SA) (see Section 4.3 for more details). The objective function to be minimized is defined as:

$$\Upsilon = \mathbb{E}\left[\hat{\mathtt{N}}\right] + f_p(\mathsf{P}_\mathsf{F}^{\mathrm{up}})$$

where $\mathbb{E}\left[\hat{\mathtt{N}}\right]$ is the estimated number of inactivations needed for decoding the surrogate LT code and f_p is defined as

$$f_p(\mathsf{P}_\mathsf{F}^{\mathrm{up}}) = \begin{cases} 0, & \mathsf{P}_\mathsf{F}^{\mathrm{up}} < \mathsf{P}_\mathsf{F}^* \\ b\ (1 - \mathsf{P}_\mathsf{F}^*/\mathsf{P}_\mathsf{F}^{\mathrm{up}}), & \text{else} \end{cases}$$

being P_F^* the target probability of decoding failure at $\delta = 15$, $\mathsf{P}_\mathsf{F}^{\mathrm{up}}$ its upper bound given in Theorem 1 and b a large positive number ($b = 10000$ was used in the example). The large b factor ensures that degree distributions which do not comply with the target probability of decoding failure are discarded.

The degree distribution obtained from our optimization is the following:

$$\begin{aligned}\Omega^*(\mathrm{x}) = {} & 0.0490\ \mathrm{x}^1 + 0.3535\ \mathrm{x}^2 + 0.1135\ \mathrm{x}^3 + 0.2401\ \mathrm{x}^4 \\ & + 0.1250\ \mathrm{x}^{10} + 0.1183\ \mathrm{x}^{11} + 0.0006\ \mathrm{x}^{40}.\end{aligned} \tag{5.10}$$

Figure 5.9 compares the output degree distribution of R10 Raptor codes Ω^{R10}, given in (5.9), with the degree distribution obtained from our optimization Ω^*, given in (5.10). Both distributions have the same average output degree, and in both cases the degree with maximum probability is 2. However, the distributions are quite different.

In order to compare the performance of the two Raptor codes considered Monte Carlo simulations were carried out. In order to derive the probability of decoding failure for each overhead value δ simulations were run until 200 errors were collected. To obtain the average number of inactivations, 1000 decodings were carried out for each overhead value δ.

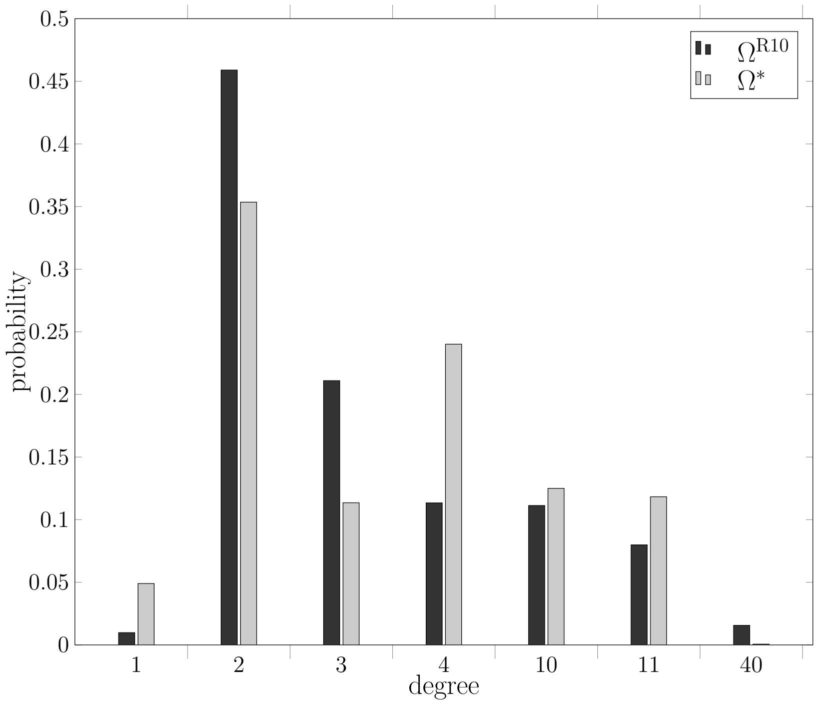


Fig. 5.9 Comparison of the output degree distribution of R10 Raptor codes, Ω^{R10} with the output degree distribution obtained through optimization Ω^{*}.

Figure 5.10 shows the probability of decoding failure $\mathsf{P_F}$ vs. the absolute receiver overhead δ for the two binary Raptor codes with Hamming outer codes, with degree distributions Ω^{R10} and Ω^{*}. The upper bound to the probability of failure is also shown for both Raptor codes. We can observe how the Raptor code with degree distribution Ω^{*} meets the design goal, its probability of decoding failure at $\delta = 15$ is below 10^{-3}. If we compare the two Raptor codes, we can see how the probability of decoding failure of the Raptor code with Ω^{*} is lower than that with Ω^{R10}. For $\mathsf{P_F}$ below 10^{-3}, the Raptor code with degree distribution Ω^{*} needs approximately 5 less overhead symbols to achieve the same $\mathsf{P_F}$ as the Raptor code with degree distribution Ω^{R10}.

In Figure 5.11 the average number of inactivations is shown as a function of the absolute receiver overhead for the two binary Raptor codes with Hamming outer codes, with degree distributions Ω^{R10} and Ω^{*}. We can observe how the degree distribution obtained from the optimization process, Ω^{*}, leads to a higher number of inactivations, and, thus, to a higher decoding complexity.

The results in Figures 5.10 and 5.11 illustrate the tradeoff between probability of decoding failure and number of inactivations (decoding complexity). In general if one

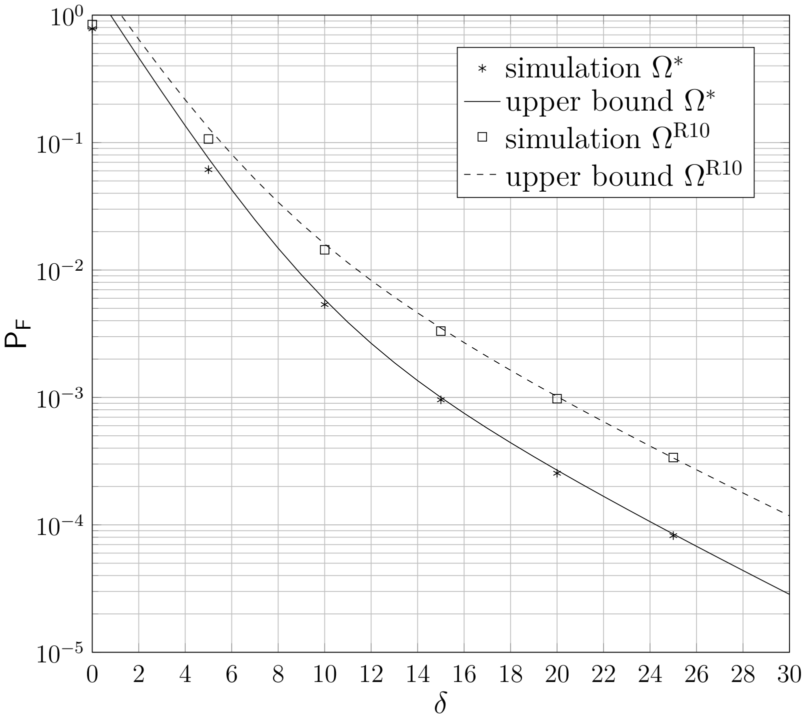


Fig. 5.10 Probability of decoding failure $\mathsf{P_F}$ vs. absolute receiver overhead δ for binary Raptor codes with a $(63, 57)$ Hamming outer code and LT degree distributions Ω^* and Ω^{R10} The markers represent the result of simulations, while the lines represents the upper bound to the probability of decoding failure in Theorem 1.

desires to improve the probability of decoding failure it is necessary to accept a higher decoding complexity.

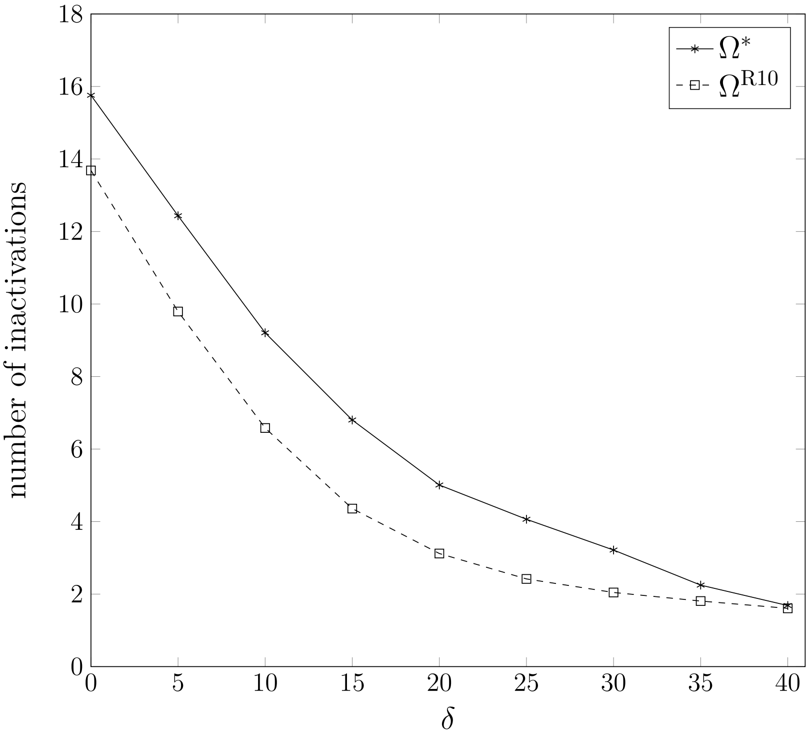


Fig. 5.11 Number of inactivations vs. absolute receiver overhead δ for binary Raptor codes with a $(63, 57)$ Hamming outer code and LT degree distributions Ω^* and Ω^{R10}.

5.4 Summary

In this chapter we have focused on Raptor codes under inactivation decoding. In Section 5.1 an upper bound to the probability of decoding failure of q-ary Raptor codes under ML decoding has been derived. This bound is based on the weight enumerator of the outer code, or its average weight enumerator when the outer code is randomly drawn from a code ensemble. The bounds derived are tight, specially in the error floor region, as it is shown by means of simulations. In Section 5.2 a heuristic method is presented that yields an approximate analysis of Raptor codes under inactivation decoding. The method is shown to be accurate for several examples. Finally in Section 5.3 a code design is presented based on the results presented in this chapter. More concretely, we have designed the degree distribution of the LT component of a binary Raptor code with a $(63, 57)$ Hamming outer code. The design goal was obtaining a probability of decoding failure $\mathsf{P_F} < 10^{-3}$ at $\delta = 15$ while minimizing the number of inactivations needed for decoding.

Chapter 6

Fixed-Rate Raptor Codes

Despite the fact that Raptor codes were originally designed for a rateless setting, they are sometimes used as fixed-rate codes due to their excellent performance and low complexity (see Section 3.3.2). In this chapter we focus on the performance of Raptor codes under ML decoding in a fixed-rate setting. More concretely we analyze the distance properties of an ensemble of (fixed-rate) Raptor codes with linear random outer codes that resembles R10 Raptor codes. This chapter is organized as follows. In Section 6.1 we introduce the ensemble of raptor codes to be studied. In Section 6.2 the average distance spectrum of the ensemble is derived. Section 6.3 presents sufficient and necessary conditions for the ensembles of Raptor codes to have a minimum distance growing linearly with the block length. In Section 6.4 simulations are presented that validate the analytical results obtained in this chapter. Moreover, it is shown by means of simulations how the erasure correcting properties of the ensemble studied in this chapter resemble those of standard R10 Raptor codes as a first order approximation. Finally, the main contributions of this chapter are summarized in Section 6.5.

6.1 Raptor Code Ensemble

A fixed-rate Raptor can be seen as the concatenation of a precode with a fixed-rate LT code, as shown in Figure 6.1.

In general analyzing the distance properties of one particular code is very difficult. In [45] it was shown how the problem of finding the weights of a linear code is NP complete, that is, no fast solution to this problem is known. Instead of focusing on one particular Raptor code we will focus on an *ensemble* of Raptor codes and derive average results for this ensemble. We focus on Raptor code ensembles where the

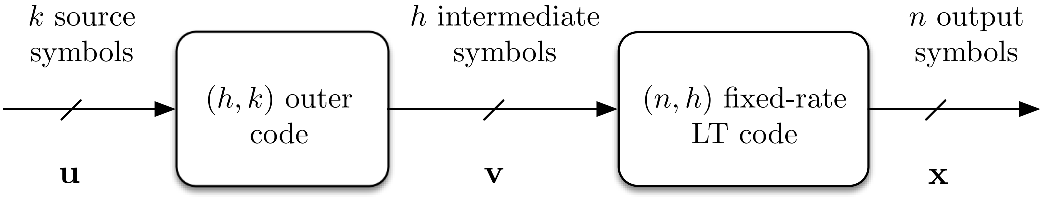


Fig. 6.1 A fixed-rate Raptor code consists of a serial concatenation of a linear block code (pre-code) with a fixed-rate LT code.

outer code belongs to the linear random ensemble. The choice of this ensemble is not arbitrary. The outer code used by the R10 Raptor code, the most widespread version of binary Raptor codes (see Section 3.3.2), is a concatenation of two systematic codes, the first being a high-rate regular LDPC code and the second a pseudo-random code characterized by a dense parity check matrix. The outer codes of R10 Raptor codes were designed to behave as codes drawn from the linear random ensemble in terms of rank properties, but allowing a fast algorithm for matrix-vector multiplication [60]. Thus, the ensemble we analyze may be seen as a simple model for practical Raptor codes with outer codes specifically designed to mimic the behavior of linear random codes. This model has the advantage to make the analytical investigation tractable. Moreover, in spite of its simplicity, this model provides us an insight into the behaviour of R10 Raptor codes in a fixed-rate setting, as illustrated by simulation results in this chapter.

The ensemble of Raptor codes we will analyze is obtained by a serial concatenation of an outer code in the $(\mathsf{r_i} n, \mathsf{r_o}\mathsf{r_i} n)$ binary linear random block code ensemble $\mathscr{C}_\mathsf{o}$[1], with all possible realizations of an $(n, \mathsf{r_i} n)$ fixed-rate LT code with output degree distribution $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \ldots, \Omega_{d_{\max}}\}$. We denote this ensemble as $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$.

In our analysis we often talk about expected properties of a code selected randomly in the ensemble $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$. This random selection is performed first by randomly drawing the parity-check matrix of the linear random precode. This is achieved drawing $(h-k)h$ independent and identically distributed Bernoulli uniform random variables, each of which is associated to one element of the parity check matrix. Second, the LT code is generated according to the usual LT encoding process. Each output symbol is generated independently from all other symbols by drawing a degree d according to Ω

[1] This ensemble was first analyzed by Gallager in his PhD Thesis [16] and is sometimes known as the Gallager random code ensemble.

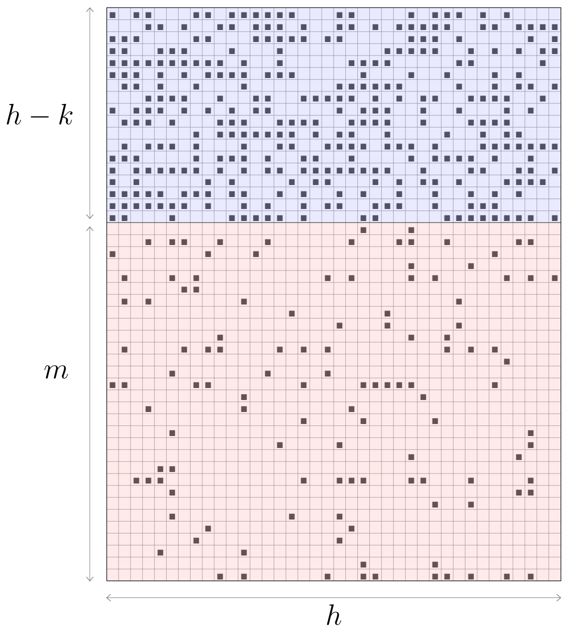


Fig. 6.2 Constraint matrix of a Raptor code with a linear random precode, with $k = 20$, $h = 38$ and $m = 30$. The blue sub-matrix represents the parity check matrix of the precode. The red sub-matrix represents the transposed generator matrix of the LT code.

and then choosing uniformly at random d distinct symbols out of the h intermediate ones.

For illustration in Figure. 6.2 we provide the constraint matrix for a Raptor code with a linear random precode, with $k = 20$, $h = 38$ and $m = 30$ with the LT degree distribution of R10 Raptor codes. In the upper part, highlighted in blue, the parity check matrix of the precode code can be distinguished. It can be observed how this sub-matrix is dense. The lower part of the constraint matrix (highlighted in red) corresponds to the LT symbols and is sparse. If we compare this constraint matrix with the constraint matrix of R10 Raptor codes in Figure 3.16, we can see how the parity check matrix of the outer code is now considerably denser. Hence, if we were to use a Raptor code with a linear random precode in practice, in general decoding would be more complex. For example, if we would use inactivation decoding we would need in general more inactivations for decoding.

A related ensemble was analyzed in [75], where lower bounds on the distance and error exponent are derived for a concatenated scheme with random outer code and a fixed inner code.

6.2 Distance Spectrum

In this section the expected weight enumerator (WE) of a fixed-rate Raptor code picked randomly in the ensemble $\mathscr{C}(\mathscr{C}_{\mathsf{o}}, \Omega, \mathsf{r}_{\mathsf{i}}, \mathsf{r}_{\mathsf{o}}, n)$ is characterized. We first obtain the expression for the expected weight enumerator. Then, we analyze the asymptotic exponent of the weight enumerator.

Theorem 4. *Let A_d be the expected multiplicity of codewords of weight d for a code picked randomly in the ensemble $\mathscr{C}(\mathscr{C}_{\mathsf{o}}, \Omega, \mathsf{r}_{\mathsf{i}}, \mathsf{r}_{\mathsf{o}}, n)$. For $d \geq 1$ we have*

$$A_d = \binom{n}{d} 2^{-h(1-\mathsf{r}_{\mathsf{o}})} \sum_{l=1}^{h} \binom{h}{l} p_l^d (1-p_l)^{n-d} \tag{6.1}$$

where

$$p_l = \sum_{j=1}^{d_{\max}} \Omega_j \sum_{\substack{i=\max(1,l+j-h)\\ i \text{ odd}}}^{\min(l,j)} \frac{\binom{j}{i}\binom{h-j}{l-i}}{\binom{h}{l}} = \sum_{j=1}^{d_{\max}} \Omega_j \sum_{\substack{i=\max(1,l+j-h)\\ i \text{ odd}}}^{\min(l,j)} \frac{\binom{l}{i}\binom{h-l}{j-i}}{\binom{h}{j}}.$$

Proof. For serially concatenated codes we have

$$A_d = \sum_{l=1}^{h} \frac{A_l^{\mathsf{o}} A_{l,d}^{\mathsf{i}}}{\binom{h}{l}}, \tag{6.2}$$

being A_l^{o} the average weight enumerator of the outer code, and $A_{l,d}^{\mathsf{i}}$ the average input output-weight enumerator function of the inner (fixed-rate) LT code. For an (h,k) linear random code, the average weight enumerator is known to be [16]

$$A_l^{\mathsf{o}} = \binom{h}{l} 2^{-h(1-\mathsf{r}_{\mathsf{o}})}. \tag{6.3}$$

Let us now focus on the average input output-weight enumerator function of the fixed-rate LT code. We denote by l the Hamming weight of the input word to the LT encoder and by $p_{j,l}$ the probability that a randomly chosen output symbol generated by the LT encoder takes the value 1 given that the intermediate word has Hamming weight l and the degree of the LT code output symbol is j, i.e.,

$$p_{j,l} := \Pr\{X_i = 1 | w_{\mathsf{H}}(\mathbf{V}) = l, \deg(X_i) = j\}$$

for any $i \in \{1, \dots, n\}$. We may express this probability as

$$p_{j,l} = \sum_{\substack{i=\max(1,l+j-h)\\ i \text{ odd}}}^{\min(l,j)} \frac{\binom{j}{i}\binom{h-j}{l-i}}{\binom{h}{l}} = \sum_{\substack{i=\max(1,l+j-h)\\ i \text{ odd}}}^{\min(l,j)} \frac{\binom{l}{i}\binom{h-l}{j-i}}{\binom{h}{j}}$$

By removing the conditioning on j, p_l is obtained, that is the probability of any of the n output bits of the fixed-rate LT encoder taking value 1 given a Hamming weight l for the intermediate word, i.e.,

$$p_l := \Pr\{X_i = 1 | w_{\mathsf{H}}(\mathbf{V}) = l\}$$

for any $i \in \{1, \dots, n\}$. We have

$$p_l = \sum_{j=1}^{d_{\max}} \Omega_j p_{j,l}. \tag{6.4}$$

Given the fact that every output bit is generated independently, the Hamming weight of the LT codeword conditioned to an intermediate word of weight l is a binomially distributed random variable with parameters n and p_l. Hence,

$$\Pr\{w_{\mathsf{H}}(\mathbf{X}) = d | w_{\mathsf{H}}(\mathbf{V}) = l\} = \binom{n}{d} p_l^d (1 - p_l)^{n-d}. \tag{6.5}$$

We are now in the position of calculating the average input output-weight enumerator function of a LT code by multiplying (6.5) by the number of weight-l intermediate words, yielding

$$A^{\mathsf{i}}_{l,d} = \binom{h}{l}\binom{n}{d} p_l^d (1 - p_l)^{n-d}. \tag{6.6}$$

Finally, by making use of (6.2), (6.3) and (6.6), we obtain (6.1). □

Corollary 1. *As opposed to A_d with $d \geq 1$, whose expression is given in Theorem 4, the expected number of codewords of weight 0, A_0, is given by*

$$\begin{aligned} A_0 &= 1 + \sum_{l=1}^{h} \frac{A^{\mathsf{o}}_l A^{\mathsf{i}}_{l,0}}{\binom{h}{l}} \\ &= 1 + 2^{-n\mathsf{r}_{\mathsf{i}}(1-\mathsf{r}_{\mathsf{o}})} \sum_{l=1}^{h} \binom{h}{l} (1 - p_l)^n . \end{aligned}$$

Remark 3. *An expected number of weight-*0 *codewords larger than one implies that different input messages can be mapped into the same codeword. The fact that* $A_0 > 1$ *is because we are drawing both the outer and inner code at random. Hence, we do not ensure by construction that different input messages are mapped into different codewords. In Section 6.3, Theorem 8, it will be shown that if the* $(\mathsf{r_i}, \mathsf{r_o})$ *pair belongs to a region that is refereed to as "positive normalized typical minimum distance region", the expected number* A_0 *of zero weight codewords approaches* 1 *(exponentially) as* n *increases.*

So far we have considered finite length Raptor codes. Often, when dealing with ensembles of codes, their distance properties can be captured in a very compact form by letting the block length n tend to infinity, while keeping the code rate constant. Such analysis of LDPC codes was performed by Gallager in his Ph.D. Thesis [16]. Hereafter, we denote the normalized output weight of the fixed-rate Raptor code by $\varpi = d/n$ and the normalized output weight of the outer code (input weight to the LT encoder) by $\lambda = l/h$. The asymptotic exponent of the weight distribution of an ensemble is defined as

$$G(\varpi) = \lim_{n\to\infty} \frac{1}{n} \log_2 \mathcal{A}_{\varpi n} .$$

Commonly, $G(\varpi)$ is also referred to as growth rate. The growth rate of a code or code ensemble is a compact representation of the properties of the code when its block length is asymptotically large. In particular, if for a given normalized output weight ϖ, we have $G(\varpi) > 0$, we expect to have asymptotically many codewords with normalized weight ϖ. On the other hand, if for a given ϖ we have $G(\varpi) < 0$, we expect to have asymptotically few codewords with weight ϖ.

Next we compute the growth rate of the weight distribution for the ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$, that is the ensemble $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$ in the limit where n tends to infinity for constant $\mathsf{r_i}$ and $\mathsf{r_o}$.

Theorem 5. *The asymptotic exponent of the weight distribution of the fixed-rate Raptor code ensemble* $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$ *is given by*

$$G(\varpi) = \mathsf{H_b}(\varpi) - \mathsf{r_i}(1 - \mathsf{r_o}) + \mathsf{f}_{\max}(\varpi) \tag{6.7}$$

where $\mathsf{H_b}$ *is the binary entropy function and*

$$\mathsf{f}_{\max}(\varpi) := \max_{\lambda \in \mathscr{D}_\lambda} \mathsf{f}(\varpi, \lambda),$$

being $\mathsf{f}(\varpi, \lambda)$ *and* $\mathscr{D}_\lambda$ *defined as follows,*

$$\mathsf{f}(\varpi, \lambda) := \mathsf{r_i}\mathsf{H_b}(\lambda) + \varpi \log_2 \varrho_\lambda + (1-\varpi)\log_2\left(1-\varrho_\lambda\right) \,,$$

$$\mathscr{D}_\lambda = \begin{cases} (0,1) & \textit{if}\ \Omega_j = 0\ \textit{for all even}\ j \\ (0,1] & \textit{otherwise}\,, \end{cases}$$

and with ϱ_λ *defined as*

$$\varrho_\lambda := \frac{1}{2}\sum_{j=1}^{d_{\max}} \Omega_j \left[1 - (1-2\lambda)^j\right].$$

Proof. Let us define $\mathbb{N}_h^* = \{1, 2, \ldots, h\}$. From (6.1) we have

$$\begin{aligned}
&\frac{1}{n}\log_2 A_{\varpi n}\\
&= \frac{1}{n}\log_2\binom{n}{\varpi n} - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2\sum_{l=1}^{h}\binom{h}{l}p_l^d(1-p_l)^{n-d}\\
&\overset{\text{(a)}}{\le} \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2\left(2\pi n\varpi(1-\varpi)\right) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2\sum_{l=1}^{h}\binom{h}{l}p_l^d(1-p_l)^{n-d}\\
&\overset{\text{(b)}}{\le} \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2\left(2\pi n\varpi(1-\varpi)\right) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n)\\
&+ \frac{1}{n}\log_2\max_{l\in\mathbb{N}_{h-1}^*}\left\{\binom{h}{l}p_l^d(1-p_l)^{n-d}\right\}\\
&\overset{\text{(c)}}{\le} \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2(2\pi n\varpi(1-\varpi)) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n)\\
&+ \max_{l\in\mathbb{N}_{h-1}^*}\left\{\mathsf{r_i}\mathsf{H_b}\left(\frac{l}{h}\right) - \frac{1}{2n}\log_2\left(2\pi\mathsf{r_i}n\frac{l}{h}\left(1-\frac{l}{h}\right)\right)\right.\\
&\left.+\,\varpi\log_2 p_l + (1-\varpi)\log_2(1-p_l)\right\}\\
&= \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2(2\pi n\varpi(1-\varpi)) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n)\\
&+ \max_{\lambda\in\left\{\frac{1}{\mathsf{r_i}n},\ldots,\frac{\mathsf{r_i}n-1}{\mathsf{r_i}n}\right\}}\left\{\mathsf{r_i}\mathsf{H_b}\left(\lambda\right) - \frac{1}{2n}\log_2\left(2\pi\mathsf{r_i}n\lambda\left(1-\lambda\right)\right) + \varpi\log_2 p_{\mathsf{r_i}n\lambda}\right.\\
&\left.+\,(1-\varpi)\log_2(1-p_{\mathsf{r_i}n\lambda})\right\}
\end{aligned}\tag{6.8}$$

Inequality (a) follows from the well-known tight bound [16]

$$\binom{n}{\sigma n} \le \frac{2^{n\mathsf{H_b}(\sigma)}}{\sqrt{2\pi n\sigma(1-\sigma)}}, \qquad 0<\sigma<1 \tag{6.9}$$

where $\mathsf{H_b}$ is the binary entropy function, while (b) follows from

$$\sum_{l=1}^{h}\binom{h}{l}p_l^d(1-p_l)^{n-d} \le h\max_{l\in\mathbb{N}_h^*}\binom{h}{l}p_l^d(1-p_l)^{n-d}$$

and from the fact that the maximum cannot be taken for $l = h$ for large enough n (as shown next). Inequality (c) is due again to (6.9), to $\log_2(\cdot)$ being a monotonically increasing function, and to $1/n$ being a scaling factor not altering the result of the maximization with respect to l.

We may prove the fact that the maximum is not taken for $l = h$, for large enough h, as follows. By calculating directly (6.4) for $l = h$ and $l = h - 1$ it is easy to show that we have

$$p_h = \sum_{\substack{j=1 \\ j \text{ odd}}}^{d_{\max}} \Omega_j \, ,$$

and

$$p_{h-1} = \sum_{\substack{j=1 \\ j \text{ odd}}}^{d_{\max}} \frac{h-j}{h} \Omega_j + \sum_{\substack{j=1 \\ j \text{ even}}}^{d_{\max}} \frac{j}{h} \Omega_j \, .$$

For increasing h we have $p_{h-1}/p_h \to 1$. Hence, there exists $h_0(\Omega)$ such that

$$h \, p_{h-1}^d (1 - p_{h-1})^{n-d} > p_h^d (1 - p_h)^{n-d}$$

for all $h > h_0(\Omega)$. Hence, for all such values of h the maximum cannot be taken at $l = h$.

Next, by defining

$$\begin{aligned} \hat{\lambda}_n = \operatorname*{argmax}_{\lambda \in \left\{ \frac{1}{\mathsf{r_i} n}, \frac{2}{\mathsf{r_i} n}, \dots, \frac{\mathsf{r_i} n - 1}{\mathsf{r_i} n} \right\}} & \Bigg\{ \mathsf{r_i} H_b(\lambda) - \frac{1}{2n} \log_2 (2\pi \mathsf{r_i} n \lambda (1 - \lambda)) \\ & + \varpi \log_2 p_{\mathsf{r_i} n \lambda} + (1 - \varpi) \log_2 (1 - p_{\mathsf{r_i} n \lambda}) \Bigg\} \end{aligned}$$

the right-hand side of (6.8) may be recast as

$$\begin{aligned} & \mathsf{H_b}(\varpi) - \frac{1}{2n} \log_2 \left(2\pi n \varpi (1 - \varpi) \right) - \mathsf{r_i}(1 - \mathsf{r_o}) + \frac{1}{n} \log_2 (\mathsf{r_i} n) \\ & + \mathsf{r_i} \mathsf{H_b}(\hat{\lambda}_n) - \frac{1}{2n} \log_2 (2\pi \mathsf{r_i} n \hat{\lambda}_n (1 - \hat{\lambda}_n)) + \varpi \log_2 p_{\mathsf{r_i} n \hat{\lambda}_n} + (1 - \varpi) \log_2 (1 - p_{\mathsf{r_i} n \hat{\lambda}_n}) \, . \end{aligned}$$

The two terms $\frac{1}{2n} \log_2 (2\pi n \varpi (1 - \varpi))$ and $\frac{1}{n} \log_2 (\mathsf{r_i} n)$ in the last expression converge to zero as $n \to \infty$. Moreover, also the term $\frac{1}{2n} \log_2 (2\pi \mathsf{r_i} n \hat{\lambda}_n (1 - \hat{\lambda}_n))$ converges to zero regardless of the behavior of the sequence $\hat{\lambda}_n$. In fact, it is easy to check that the term $\frac{1}{2n} \log_2 (2\pi \mathsf{r_i} n \hat{\lambda}_n (1 - \hat{\lambda}_n))$ converges to zero in the limiting cases $\hat{\lambda}_n = \frac{1}{\mathsf{r_i} n}$ $\forall n$ and $\hat{\lambda}_n = \frac{\mathsf{r_i} n - 1}{\mathsf{r_i} n}$ $\forall n$, so it does in all other cases.

If we now develop the right hand side of (6.8) further, for large enough n, we have

$$\begin{aligned}
&\mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2(2\pi n\varpi(1-\varpi)) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n) \\
&+ \max_{\lambda\in\left\{\frac{1}{\mathsf{r_i}n},\dots,\frac{\mathsf{r_i}n-1}{\mathsf{r_i}n}\right\}} \left\{\mathsf{r_i}\mathsf{H_b}(\lambda) - \frac{1}{2n}\log_2(2\pi\mathsf{r_i}n\lambda(1-\lambda)) + \varpi\log_2 p_{\mathsf{r_i}n\lambda}\right. \qquad (6.10)\\
&\left.+ (1-\varpi)\log_2(1-p_{\mathsf{r_i}n\lambda})\right\} \\
&\overset{\text{(d)}}{\le} \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2(2\pi n\varpi(1-\varpi)) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n) \\
&+ \sup_{\lambda\in\mathbb{Q}\cap(0,1)} \left\{\mathsf{r_i}\mathsf{H_b}(\lambda) - \frac{1}{2n}\log_2(2\pi\mathsf{r_i}n\lambda(1-\lambda)) + \varpi\log_2\left(\varrho_\lambda + \frac{K}{n}\right)\right. \\
&\qquad\qquad \left.+ (1-\varpi)\log_2\left(1-\varrho_\lambda + \frac{K}{n}\right)\right\} \\
&\overset{\text{(e)}}{=} \mathsf{H_b}(\varpi) - \frac{1}{2n}\log_2(2\pi n\varpi(1-\varpi)) - \mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n) \\
&+ \sup_{\lambda\in(0,1)} \left\{\mathsf{r_i}\mathsf{H_b}(\lambda) - \frac{1}{2n}\log_2(2\pi\mathsf{r_i}n\lambda(1-\lambda)) + \varpi\log_2\left(\varrho_\lambda + \frac{K}{n}\right)\right. \\
&\qquad\qquad \left.+ (1-\varpi)\log_2\left(1-\varrho_\lambda + \frac{K}{n}\right)\right\} \\
&:= \Gamma_n(\varpi).
\end{aligned}$$

being $\mathbb{Q}$ the set of rational numbers. Inequality (d) follows from the fact that, as it can be shown, $|\varrho_\lambda - p_{\mathsf{r_i}n\lambda}| < K/n$ (uniformly in λ) for large enough n and from the fact that the supremum over $\mathbb{Q}\cap(0,1)$ upper bounds the maximum over the finite set $\left\{\frac{1}{\mathsf{r_i}n},\dots,\frac{\mathsf{r_i}n-1}{\mathsf{r_i}n}\right\}$. Equality (e) is due to the density of $\mathbb{Q}$. In equality (e), the function of λ being maximized is regarded as a function over the real interval $(0,1)$ (i.e., λ is regarded as a real parameter).

The upper bound (6.10) on $\frac{1}{n}\log_2 A_{\varpi n}$ is valid for any finite but large enough n. If we now let n tend to infinity, all inequalities (a)–(d) are satisfied with equality. In particular: for (a) this follows from the well-known exponential equivalence $\binom{n}{\varpi n} \doteq 2^{n\mathsf{H_b}(\varpi)}$; for (b) from the exponential equivalence $\sum_l 2^{nf(l)} \doteq \max_l 2^{nf(l)}$; for (c) from $\binom{\mathsf{r_i}n}{\hat{\lambda}_n\mathsf{r_i}n} \doteq 2^{n\mathsf{H_b}(\hat{\lambda}_n)}$ (due to $\frac{1}{2n}\log_2(2\pi\mathsf{r_i}n\hat{\lambda}_n(1-\hat{\lambda}_n))$ vanishing for large n); for (d) from the fact that, asymptotically in n, applying the definition of limit we can show that the maximum over the set $\left\{\frac{1}{\mathsf{r_i}n},\dots,\frac{\mathsf{r_i}n-1}{\mathsf{r_i}n}\right\}$ upper bounds the supremum over $\mathbb{Q}\cap(0,1)$ (while at the same time being upper bounded by it for any n). The expression of ϱ_λ is obtained by assuming n tending to ∞ using the expression of p_l. Alternatively, the same expression is obtained by assuming n tending to ∞ and letting an output symbol of degree i choose its i neighbors *with* replacement.

By letting n tend to infinity and by cancelling all vanishing terms, we finally obtain the statement. Note that we can replace the supremum by a maximum over $\mathscr{D}_\lambda$ as this maximum is always well-defined.[2] $\square$

In the next two lemmas, which will be useful in the sequel, the derivative of the growth rate function is characterized. For the sake of clarity, we introduce the notation $\varrho(\lambda)$ instead of ϱ_λ. Hence, we stress the fact that $\varrho(\lambda)$ is a function of λ.

Lemma 1. *The derivative of the growth rate of the weight distribution of a fixed-rate Raptor code ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r}_\mathsf{i}, \mathsf{r}_\mathsf{o})$ is given by*

$$G'(\varpi) = \log_2 \frac{1-\varpi}{\varpi} + \log_2 \frac{\varrho(\lambda_0)}{1-\varrho(\lambda_0)}$$

where

$$\lambda_0(\varpi) := \arg\max_{\lambda \in \mathscr{D}_\lambda} \left\{ \mathsf{f}(\varpi, \lambda) \right\} . \tag{6.11}$$

Proof. Let us rewrite the expression of $G(\varpi)$ in (6.7) as

$$G(\varpi) = \mathsf{H}_\mathsf{b}(\varpi) - \mathsf{r}_\mathsf{i}(1-\mathsf{r}_\mathsf{o}) + \mathsf{f}(\varpi, \lambda_0(\varpi)) .$$

We must have

$$\frac{\partial \mathsf{f}}{\partial \lambda}(\varpi, \lambda_0) = 0 . \tag{6.12}$$

Where λ_0 is actually a function of ϖ, $\lambda_0(\varpi)$. Taking the derivative with respect to ϖ, and after elementary algebraic manipulation we obtain

$$G'(\varpi) = \log_2 \frac{1-\varpi}{\varpi} + \log_2 \frac{\varrho(\lambda_0)}{1-\varrho(\lambda_0)} + \frac{\partial \mathsf{f}}{\partial \lambda}(\varpi, \lambda_0) \frac{\mathrm{d}\lambda_0}{\mathrm{d}\varpi}$$

which, applying (6.12), yields the statement. $\square$

[2] In fact, for any $\varpi \in [0,1]$ the function $\mathsf{f}(\varpi, \lambda)$ diverges to $-\infty$ as $\lambda \to 0^+$. Moreover, it diverges to $-\infty$ as $\lambda \to 1^-$ if $\Omega_j = 0$ for all even j and converges as $\lambda \to 1^-$ otherwise. Finally, for all $\varpi \in [0,1]$ it is continuous for all $\lambda \in \mathscr{D}_\lambda$.

Lemma 2. *For all $0 < \varpi < 1/2$, the derivative of the growth rate of the weight distribution of a fixed-rate Raptor code ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r}_\mathsf{i}, \mathsf{r}_\mathsf{o})$ fulfills*

$$G'(\varpi) > 0.$$

Proof. If in the expression for $G'(\varpi)$ in Lemma 1 we impose $G'(\varpi) = 0$, we obtain

$$\frac{1-\varpi}{\varpi} = \frac{1-\varrho(\lambda_0)}{\varrho(\lambda_0)}.$$

Since the function $(1-x)/x$ is monotonically decreasing for $x \in (0,1)$, this implies $\varpi = \varrho(\lambda_0)$. Next, observing (6.11), by the definition of λ_0, its partial derivative $\partial \mathsf{f}(\varpi, \lambda)/\partial \lambda$ must be zero when calculated for $\lambda = \lambda_0$. The expression of this partial derivative is

$$\frac{\partial \mathsf{f}}{\partial \lambda}(\varpi, \lambda) = \mathsf{r}_\mathsf{i} \log_2 \frac{1-\lambda}{\lambda} + \frac{\varrho'(\lambda)}{\log 2} \cdot \frac{\varpi - \varrho(\lambda)}{\varrho(\lambda)(1-\varrho(\lambda))}$$

so we obtain

$$\mathsf{r}_\mathsf{i} \log_2 \frac{1-\lambda_0}{\lambda_0} + \frac{\varrho'(\lambda_0)}{\log 2} \cdot \frac{\varpi - \varrho(\lambda_0)}{\varrho(\lambda_0)(1-\varrho(\lambda_0))} = 0\,.$$

As shown above, for any ϖ such that $G'(\varpi) = 0$ we have $\varpi = \varrho(\lambda_0)$. Substituting in the latter equation we obtain $\lambda_0 = 1/2$ which implies $\varpi = \varrho(1/2) = 1/2$. Therefore, the only value of ϖ such that $G'(\varpi) = 0$ is $\varpi = 1/2$. Due to continuity of $G'(\varpi)$ and to the fact that $G'(\varpi) \to +\infty$ as $\varpi \to 0^+$ (as shown in Appendix B.2.2). Therefore, we conclude that $G'(\varpi) > 0$ for all $0 < \varpi < 1/2$. □

A useful concept when characterizing the distance properties of an ensemble is the (normalized) typical minimum distance, which we define formally as follows.

Definition 13. *The normalized typical minimum distance of a fixed-rate Raptor code ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r}_\mathsf{i}, \mathsf{r}_\mathsf{o})$ is the real number*

$$\varpi^\star := \begin{cases} 0 & \text{if } \lim_{\varpi \to 0^+} G(\varpi) \geq 0 \\ \inf\{\varpi > 0 : G(\varpi) > 0\} & \text{otherwise.} \end{cases}$$

Example 2. *Figure 6.3 shows the growth rate $G(\varpi)$ for three different ensembles $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega^{\mathrm{R10}}, \mathsf{r}_\mathsf{i}, \mathsf{r}_\mathsf{o})$, where Ω^{R10} is the output degree distribution used in the standards [30], [31] (see details in Table 6.1) and $\mathsf{r}_\mathsf{o} = 0.99$ for three different r_i values. The growth rate of a linear random code ensemble with rate $\mathsf{r} = 0.99$ is also shown. It can be observed how the curve for $\mathsf{r}_\mathsf{i} = 0.95$ does not cross the x-axis, the curve for $\mathsf{r}_\mathsf{i} = 0.88$*

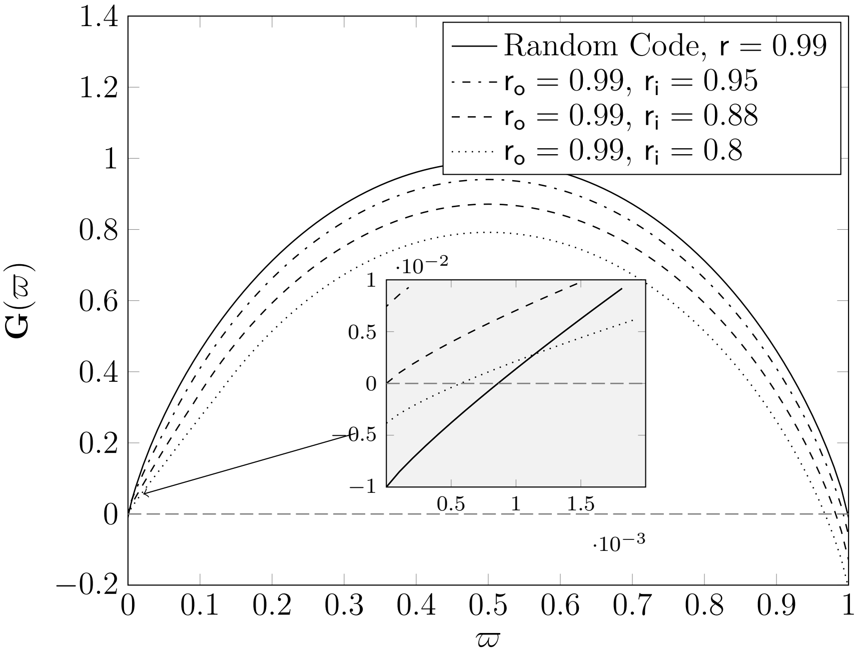


Fig. 6.3 Growth rate vs. normalized output weight ϖ. The solid line shows the growth rate of a linear random code with rate $\mathsf{r} = 0.99$. The dot-dashed, dashed, and dotted lines show the growth rates $G(\varpi)$ of the ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega^{\mathrm{R10}}, \mathsf{r_i}, \mathsf{r_o} = 0.99)$ for $\mathsf{r_i} = 0.95$, 0.88 and 0.8, respectively.

has $\varpi^\star = 0$ and the curve for $\mathsf{r_i} = 0.8$ has $\varpi^\star = 0.0005$.

This example highlights an important fact, if we fix the outer code rate to a very high value, concretely $\mathsf{r_o} = 0.99$, our ensemble still can achieve a (normalized) typical minimum distance larger than 0 when the rate of the inner code is low enough, (in this case $\mathsf{r_i} < 0.88$).

Example 3. *Figure 6.4 shows the overall rate r of the fixed-rate Raptor code ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega^{\mathrm{R10}}, \mathsf{r_i} = \mathsf{r}/\mathsf{r_o}, \mathsf{r_o})$ versus the normalized typical minimum distance $\varpi^\star$. It can be observed how, for constant overall rate r, $\varpi^\star$ increases as the outer code rate $\mathsf{r_o}$ decreases. We can also observe how decreasing $\mathsf{r_o}$ allows to get closer to the asymptotic Gilbert-Varshamov bound. Thus, if the overall rate r is kept constant, the distance properties of the fixed-rate Raptor code ensemble improve as the rate of the outer code decreases. Note that by decreasing the outer code rate the decoding complexity will generally increase.*

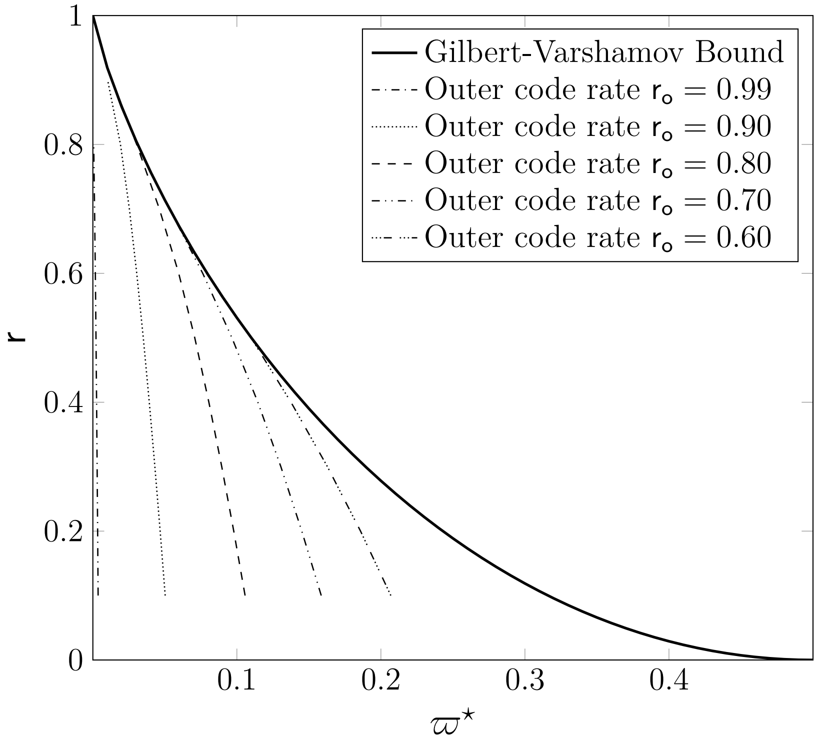


Fig. 6.4 Overall rate r vs. the normalized typical minimum distance $\varpi^\star$. The solid line represents the asymptotic Gilbert-Varshamov bound. The markers represent Raptor codes ensembles $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega^\dagger, \mathsf{r_i} = \mathsf{r}/\mathsf{r_o}, \mathsf{r_o})$ with different outer code rates, $\mathsf{r_o}$.

6.3 Positive Distance Region

The objective of this section is determining the conditions that a fixed-rate Raptor ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$ needs to fulfil in order to exhibit good normalized typical distance properties. More specifically, given a degree distribution Ω and an overall rate r for the Raptor code, we are interested in the allocation of the rate between the outer code and the fixed-rate LT code to achieve a strictly positive normalized typical minimum distance.

Definition 14 (Positive normalized typical minimum distance region)**.** *The positive normalized typical minimum distance region of an ensemble* $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$ *is defined as the set* $\mathscr{P}$ *of code rate pairs* $(\mathsf{r_i}, \mathsf{r_o})$ *for which the ensemble possesses a positive normalized typical minimum distance. Formally:*

$$\mathscr{P} := \{(\mathsf{r_i}, \mathsf{r_o}) \succeq (0,0) | \varpi^\star(\Omega, \mathsf{r_i}, \mathsf{r_o}) > 0\}$$

where we have used the notation $\varpi^\star = \varpi^\star(\Omega, \mathsf{r_i}, \mathsf{r_o})$ *to emphasize the dependence on* Ω, $\mathsf{r_i}$ *and* $\mathsf{r_o}$.

In the following theorem the positive normalized typical distance region for an LT output degree distribution Ω is developed .

Theorem 6. *The region* $\mathscr{P}$ *is given by*

$$\mathscr{P} := \left\{ (\mathsf{r_i}, \mathsf{r_o}) \succeq (0,0) | \mathsf{r_i}(1-\mathsf{r_o}) > \max_{\lambda \in \mathscr{D}_\lambda} \left\{ \mathsf{r_i} \mathsf{H_b}(\lambda) + \log_2 (1 - \varrho_\lambda) \right\} \right\} . \tag{6.13}$$

Proof. See Appendix B.2. □

The next two theorems characterize the distance properties of a fixed-rate Raptor code with linear random outer code picked randomly in the ensemble $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$ with $(\mathsf{r_i}, \mathsf{r_o})$ belonging to $\mathscr{P}$.

Theorem 7. *Let the random variable* D *be the minimum nonzero Hamming weight in the code book of a fixed-rate Raptor code picked randomly in an ensemble* $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$. *If* $(\mathsf{r_i}, \mathsf{r_o}) \in \mathscr{P}$ *then*

$$\lim_{n\to\infty} \Pr\{D \le \varpi n\} = 0$$

exponentially in n*, for all* $0 < \varpi < \varpi^\star$.

Proof. We can upper bound this probability via union bound as

$$\Pr\{D \le \varpi n\} \le \sum_{w=1}^{\varpi n} A_w .$$

We will start by proving that the sequence A_d is non-decreasing for $d < n/2$ and sufficiently large n. As $n \to \infty$, the expression $\frac{1}{n} \log_2 \frac{A_{\varpi n}}{A_{\varpi n - 1}}$ converges to $\Gamma_n(\varpi) - \Gamma_n(\varpi - \frac{1}{n})$, being $\Gamma_n(\varpi)$ given in (6.10). From Lemma 2 we know that $G'(\varpi) > 0$ for $0 < \varpi < 1/2$. Moreover, as $n \to \infty$, from Theorem 5 we have $\Gamma_n(\varpi) \to G(\varpi)$. Hence, for sufficiently large n, $\Gamma_n(\varpi) \ge \Gamma_n(\varpi - \frac{1}{n})$. This implies that A_d is non decreasing.

We can now write

$$\Pr\{D \le \varpi n\} \le \varpi n A_{\varpi n} \le \varpi n 2^{n \Gamma_n(\varpi)} ,$$

where we have used $A_{\varpi n} \le 2^{n\Gamma_n(\varpi)}$, being $\Gamma_n(\varpi)$ given in (6.10).

As $n \to \infty$ we have $\Gamma_n(\varpi) \to G(\varpi)$. Moreover, $G(\varpi) < 0$ for all $0 < \varpi < \varpi^\star$, provided $(\mathsf{r_i}, \mathsf{r_o}) \in \mathscr{P}$. Hence, $\Pr\{D \le \varpi n\}$ tends to 0 exponentially on n. □

Remark 4. *From Theorem 7, we have that when a rate point* $(\mathsf{r_i}, \mathsf{r_o})$ *belongs to the region* $\mathscr{P}$*, there is an exponential decay of the probability to find codewords with weight less than* $\varpi^\star n$*, which is a very favorable property for code ensembles. This exponential decay shall be attributed to the presence of the linear random outer code that is characterized by a dense parity-check matrix, and makes the growth rate function monotonically increasing for* ϖ *for which it is negative,* $0 < \varpi < \varpi^\star$*.*

As a comparison, for LDPC code ensembles characterized by a positive normalized typical minimum distance, the growth rate function starts from $G(0) = 0$ *with negative derivative, reaches a minimum, and then increases to cross the* x*-axis. In this case, for* $\varpi < \varpi^\star$ *the sum in the upper bound is dominated by those terms corresponding to small values of* w*, yielding either a polynomial decay (as for Gallager's codes [16]) or even* $\Pr\{D \leq \varpi n\}$ *tending to a constant (as it is for irregular unstructured LDPC ensembles [76, 77]). This is in general worse for the distance properties of the code compared to the exponential decay observed in our ensemble*

However, one should remark that this exponential decay of the probability of having codewords with weight less than $\varpi^\star n$ *comes at the cost of complexity, since the outer code is dense, and hence complex to encode and decode.*

Theorem 8. *Let the random variable* Z *be the multiplicity of codewords of weight zero in the code book of a fixed-rate Raptor code picked randomly in the ensemble* $\mathscr{C}(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o}, n)$*. If* $(\mathsf{r_i}, \mathsf{r_o}) \in \mathscr{P}$ *then*

$$\Pr\{Z > 1\} \rightarrow 0 \quad \text{as } n \rightarrow \infty\,.$$

Proof. In order to prove the statement we have to show that the probability measure of any event $\{Z = t\}$ with $t \in \mathbb{N} \setminus \{0, 1\}$ vanishes as $n \rightarrow \infty$. We start by analyzing the behavior of $\mathbb{E}[Z] = A_0$, whose expression is $\mathbb{E}[Z] = 1 + 2^{-n\mathsf{r_i}(1-\mathsf{r_o})} \sum_{l=1}^{h} \binom{h}{l} (1 - p_l)^n$. Using an argument analogous to the one adopted in the proof of Theorem 5, for large enough n we have

$$\frac{1}{n} \log_2 \left(2^{-n\mathsf{r_i}(1-\mathsf{r_o})} \sum_{l=1}^{h} \binom{h}{l} (1 - p_l)^n \right) \leq \Xi_n$$

where

$$\Xi_n := -\mathsf{r_i}(1-\mathsf{r_o}) + \frac{1}{n}\log_2(\mathsf{r_i}n) + \sup_{\lambda\in(0,1)}\Big\{\mathsf{r_i}\mathsf{H_b}(\lambda) - \frac{1}{2n}\log_2\left(2\pi\mathsf{r_i}n\lambda(1-\lambda)\right)$$
$$+\log_2(1-\varrho_\lambda + K/n)\Big\}.$$

Therefore we can upper bound $\mathbb{E}[Z]$ as $\mathbb{E}[Z] \leq 1 + 2^{n\Xi_n}$ which, if $(\mathsf{r_i},\mathsf{r_o}) \in \mathscr{P}$, implies $\mathbb{E}[Z] \to 1$ exponentially as $n \to \infty$ due to $\Xi_n \to G(0)$ and $G(0) < 0$.[3] Next, it is easy to show that $\mathbb{E}[Z] \geq 1$ and, via linear programming, that the minimum is attained if and only if $\Pr\{Z=1\} = 1$ and $\Pr\{Z=t\} = 0$ for all $t \in \mathbb{N}\setminus\{0,1\}$. Since in the limit as $n\to\infty$ of $\mathbb{E}[Z]\to 1$, we necessarily have a vanishing probability measure for any event $\{Z=t\}$ with $t\in\mathbb{N}\setminus\{0,1\}$. □

Remark 5. *From Theorem 7 and Theorem 8, a fixed-rate Raptor code picked randomly in the ensemble $\mathscr{C}(\mathscr{C}_\mathsf{o},\Omega,\mathsf{r_i},\mathsf{r_o},n)$ is characterized first by a minimum distance that is exponentially concentrated around $\varpi^\star n$ and second by an encoding function whose kernel only includes the all-zero length k message (hence bijective), with probability approaching 1 as $n\to\infty$. In other words, for rate pairs within region $\mathscr{P}$ the probability of having more than 1 weight-0 codeword tends to 0 exponentially with n. This means that the dimension of the code is k with high probability. Furthermore, for rate pairs within region $\mathscr{P}$ the minimum distance grows linearly with the block length n, which is a very favorable property.*

In the following an outer region $\mathscr{O}$ to region $\mathscr{P}$ is introduced. The outer region $\mathscr{O}$ only depends on the average output degree of the inner LT code.

Theorem 9. *The positive normalized typical minimum distance region $\mathscr{P}$ of a fixed-rate Raptor code ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o},\Omega,\mathsf{r_i},\mathsf{r_o})$ fulfills $\mathscr{P}\subseteq\mathscr{O}$, where*

$$\mathscr{O} := \left\{(\mathsf{r_i},\mathsf{r_o}) \succeq (0,0) | \mathsf{r_i} \leq \min\left(\phi(\mathsf{r_o}), \frac{1}{\mathsf{r_o}}\right)\right\}$$

with

$$\phi(\mathsf{r_o}) = \begin{cases} \frac{\bar{\Omega}\log_2(1/\mathsf{r_o})}{\mathsf{H_b}(1-\mathsf{r_o})-(1-\mathsf{r_o})} & \mathsf{r_o} > \mathsf{r_o^*} \\ 1/\mathsf{r_o} & \mathsf{r_o} \leq \mathsf{r_o^*} \end{cases}$$

[3] It is worth noting that $G(\varpi)$ is right-continuous at $\varpi = 0$. This follows from the expression of $G(\varpi)$ proved in Theorem 5 and from the fact that $\mathsf{f}_{\max}(\varpi)$ is right-continuous at $\varpi = 0$ as shown in the proof of Theorem 6.

Table 6.1 Degree distributions Ω^{R10}, defined in [30, 31] and $\Omega^{\dagger}$, defined in [28]

Degree	1	2	3	4	5	8	9	10	11	18	19	40	65	66	$\bar{\Omega}$
Ω^{R10}	0.0098	0.459	0.211	0.1134				0.1113	0.0799			0.0156			4.6314
$\Omega^{\dagger}$	0.0048	0.4965	0.1669	0.0734	0.0822	0.0575	0.036			0.0012	0.0543	0.0156	0.0182	0.0091	5.825

being $\mathsf{r}_{\mathsf{o}}^{*}$ *the only root of* $\mathsf{H}_{\mathsf{b}}(1-\mathsf{r}_{\mathsf{o}})-(1-\mathsf{r}_{\mathsf{o}})$ *in* $\mathsf{r}_{\mathsf{o}} \in (0,1)$*, numerically* $\mathsf{r}_{\mathsf{o}}^{*} \approx 0.22709$*.*

Proof. See Appendix B.3. □

Example 4. *Figures 6.5 and 6.6 show the positive normalized typical minimum distance regions,* $\mathscr{P}$ *for* Ω^{R10} *and* $\Omega^{\dagger}$ *(see Table 6.1) together with their outer bounds* $\mathscr{O}$*. We can observe how the outer bound is tight in both cases except for inner codes rates close to* $\mathsf{r}_{\mathsf{i}} = 1$*. In the figures several isorate curves are also shown, along which the rate of the Raptor code* $\mathsf{r} = \mathsf{r}_{\mathsf{i}}\ \mathsf{r}_{\mathsf{o}}$ *is constant. For example, in order to have a positive normalized typical minimum distance and an overall rate* $\mathsf{r} = 0.95$*, the figures show that the rate of the outer code must lay below* $\mathsf{r}_{\mathsf{o}} < 0.978$ *for both distributions. Let us assume for a moment we would like to design a fixed-rate Raptor code, with degree distribution* Ω^{R10} *or* $\Omega^{\dagger}$*, overall rate* $\mathsf{r} = 0.95$ *and for a given length* n*, which is assumed to be large. Different choices for* r_{i} *and* r_{o} *are possible. If* r_{o} *is not chosen as* $\mathsf{r}_{\mathsf{o}} < 0.978$ *the average minimum distance of the ensemble will not grow linearly on* n*. Hence, many codes in the ensemble will exhibit high error floors even under ML erasure decoding. This example illustrates how the concepts introduced in this chapter can be used to design a fixed-rate Raptor code. More concretely, for a constant overall rate of the Raptor code* r *we can distribute the rate among the outer code and the inner LT code such that the typical minimum distance of the Raptor code is positive (i.e., in order to have a low error floor).*

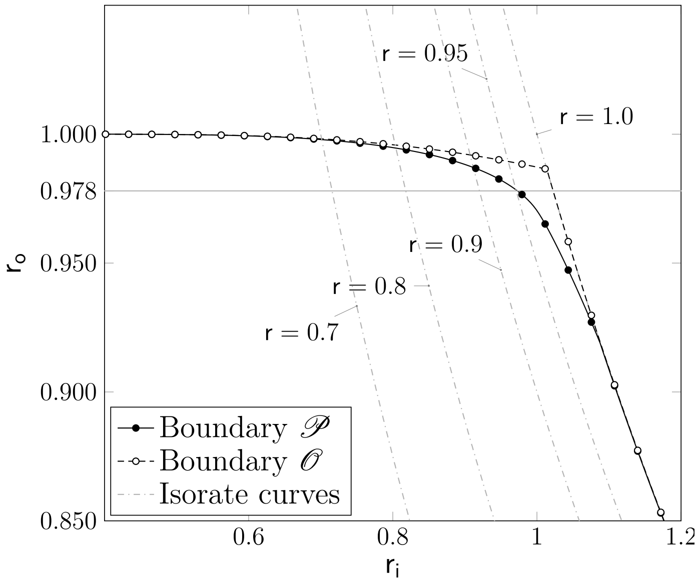


Fig. 6.5 Positive growth rate region of a fixed-rate Raptor code ensemble with degree distribution Ω^{R10}. The solid lines with black markers represent the positive growth-rate $\mathscr{P}$ and the dashed lines with white markers represents its outer bound $\mathscr{O}$. The gray dashed lines represent isorate curves for different rates r.

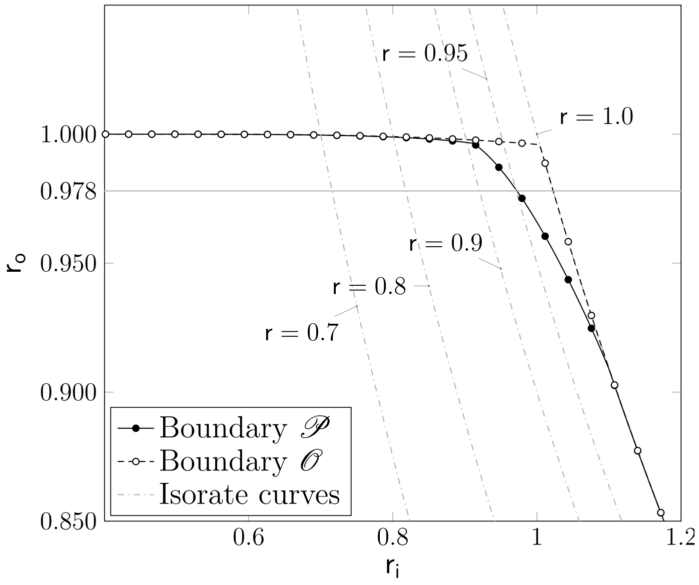


Fig. 6.6 Positive growth rate region of a fixed-rate Raptor code ensemble with degree distribution $\Omega^{\dagger}$. The solid lines with black markers represent the positive growth-rate $\mathscr{P}$ and the dashed lines with white markers represents its outer bound $\mathscr{O}$. The gray dashed lines represent isorate curves for different rates r.

6.4 Experimental Finite Length Results

In this section we present experimental results in order to validate the analytical results obtained in the previous sections. First, by means of examples it is illustrated how the developed results can be used to make accurate statements about the performance of fixed-rate Raptor code ensembles in the finite length regime. Next, we provide some results that show that a tradeoff exists between performance and decoding complexity. Finally, some simulation results are presented that illustrate that the distance properties obtained for linear random outer codes are a fair approximation for the distance properties obtained with the standard R10 Raptor outer code (see [30, 31]).

6.4.1 Results for Linear Random Outer Codes

In this section we consider Raptor code ensembles $\mathscr{C}(\mathscr{C}_{\mathsf{o}}, \Omega^{\mathrm{R10}}, \mathsf{r}_{\mathsf{i}}, \mathsf{r}_{\mathsf{o}}, n)$ for different values of r_{i}, r_{o}, and n but keeping the overall rate of the Raptor code constant to $\mathsf{r} = 0.9014$. In Figure 6.7 the boundary of $\mathscr{P}$ and $\mathscr{O}$ is shown for the LT degree distribution Ω^{R10} together with an isorate curve for $\mathsf{r} = 0.9014$. The markers placed along the isorate curve in the figure represent the two different r_{i} and r_{o} combinations that are considered in this section. The first point ($\mathsf{r}_{\mathsf{i}} = 0.9155$, $\mathsf{r}_{\mathsf{o}} = 0.9846$), marked with an asterisk, is inside but very close to the boundary of $\mathscr{P}$ for Ω^{R10}. We will refer to ensembles corresponding to this point as *bad* ensembles. The second point, ($\mathsf{r}_{\mathsf{i}} = 0.9718$, $\mathsf{r}_{\mathsf{o}} = 0.9275$) marked with a triangle, is inside but quite far from the boundary of $\mathscr{P}$ for Ω^{R10}. We will refer to ensembles corresponding to this point as *good* ensembles. In general, one would expect the good ensembles to have better distance properties than the bad ensembles, and hence, better erasure correcting properties.

Following [16] we introduce the notion of typical minimum distance for finite length, which is useful when considering ensembles of finite length Raptor codes.

Definition 15. *The typical minimum distance, $d^{\star}_{\min}$ of an ensemble $\mathscr{C}(\mathscr{C}_{\mathsf{o}}, \Omega, \mathsf{r}_{\mathsf{i}}, \mathsf{r}_{\mathsf{o}}, n)$ is defined as the integer number*

$$d^{\star}_{\min} := \begin{cases} 0 & \textit{if } A_0 > 1 + 1/2 \\ \max\{d \geq 0 : \sum_{i=0}^{d} \mathcal{A}_i - 1 < 1/2\} & \textit{otherwise.} \end{cases}$$

According to this definition, at least half of the codes in the ensemble will have a minimum distance of $d^{\star}_{\min}$ or larger. The equivalent of $d^{\star}_{\min}$ in the asymptotic regime is

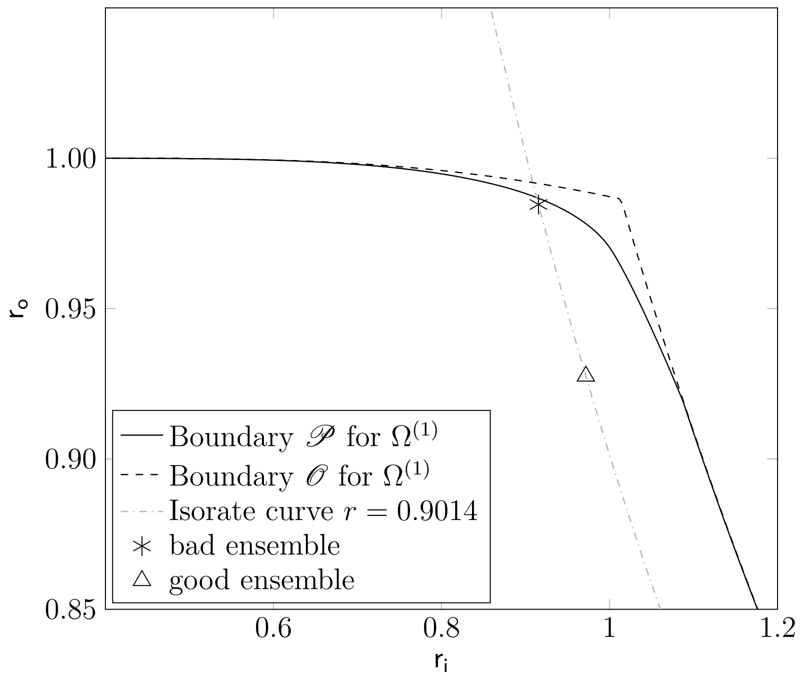


Fig. 6.7 Positive growth rate region. The solid and dashed lines represent the positive growth-rate region of Ω^{R10} its outer bound. The dashed-dotted line represents the isorate curve for $\mathsf{r} = 0.9014$ and the markers represent two different points along the isorate curve that correspond to two different code configurations with the same rate r but different values of $\mathsf{r_i}$ and $\mathsf{r_o}$. The asterisk marker represents the bad ensemble, whereas the triangle marker represents the good ensemble.

$\varpi^\star$, the (*asymptotic*) normalized minimum distance of the ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$. For sufficiently large n it is expected that $d^\star_{\min}$ converges to $\varpi^\star n$. In Figure 6.8 $d^\star_{\min}$ and $\varpi^\star n$ are shown as a function of the block length n. We can observe how the good ensemble has a larger typical minimum distance than the bad ensemble. In fact for all values of n shown in Figure 6.8 the bad ensemble has typical minimum distance $d^\star_{\min} = 0$[4]. It can also be observed how already for small values of n the $d^\star_{\min}$ and $\varpi^\star n$ are very similar. Therefore, we can say that at least for this example the result of our asymptotic analysis of the minimum distance holds already for small values of n.

The expression of the average weight enumerator in Theorem 4 can be used in order to upper bound the average codeword error rate (CER) over a BEC with erasure probability ε according to (2.3), [46]. However, the upper bound in (2.3) needs to be slightly modified to take into account codewords of weight 0. We have

[4] Since the bad ensemble is inside the positive growth rate region its minimum distance increases linearly with n. Thus for large enough n its typical minimum distance will be strictly positive.

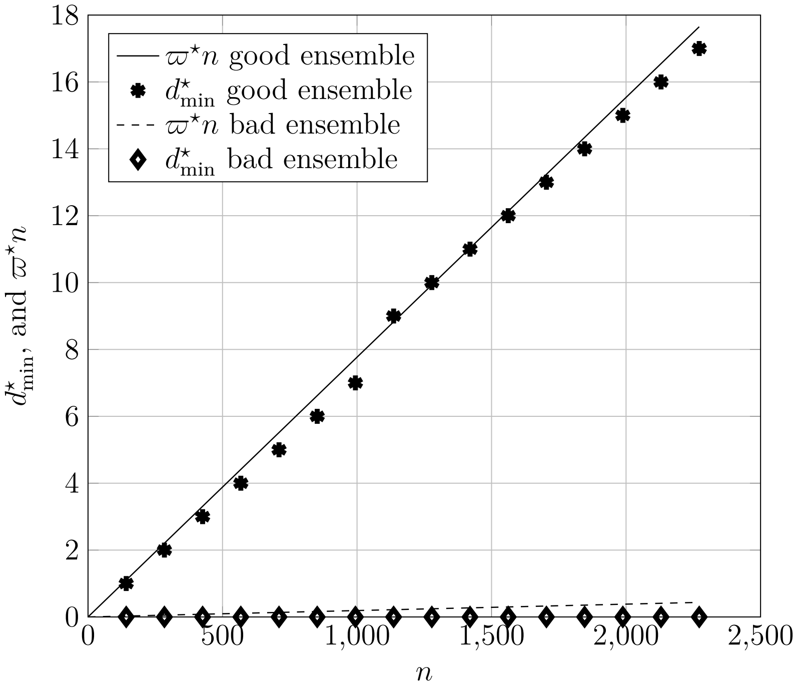


Fig. 6.8 Typical minimum distance $d^\star_{\min}$ as a function of the block length n for ensembles with $\mathsf{r_o} = 0.9275$ and $\mathsf{r_o} = 0.9846$ and $\mathsf{r} = 0.9014$. The markers represent $d^\star_{\min}$ whereas the lines represent $\varpi^\star n$.

$$\mathbb{E}_{\mathscr{C}(\mathscr{C}_\mathsf{o},\Omega,\mathsf{r_i},\mathsf{r_o},n)}\left[P_B(\varepsilon)\right] \leq P_B^{(\mathsf{S})}(n,k,\varepsilon)$$
$$+\sum_{e=1}^{n-k}\binom{n}{e}\varepsilon^e(1-\varepsilon)^{n-e}\min\left\{1,\sum_{w=1}^{e}\binom{e}{w}\frac{\mathcal{A}_w}{\binom{n}{w}}\right\}+\mathcal{A}_0-1 \tag{6.14}$$

where $P_B^{(\mathsf{S})}(n,k,\varepsilon)$ is the Singleton bound given by (2.1).

Since we consider Raptor codes in a fixed-rate setting, it is possible to expurgate Raptor code ensembles as it was done by Gallager in his PhD thesis for LDPC code ensembles [16]. Let us consider an integer $d^\star \geq 0$ so that

$$\Pr\{d_{\min} \leq d^\star\} \leq \sum_{w=0}^{d^\star} A_w - 1 = \theta < 1/2. \tag{6.15}$$

We define the expurgated ensemble $\mathscr{C}^{\text{ex}}(\mathscr{C}_\mathsf{o},\Omega,\mathsf{r_i},\mathsf{r_o},n,d^\star)$ as the ensemble composed of the codes in the ensemble $\mathscr{C}(\mathscr{C}_\mathsf{o},\Omega,\mathsf{r_i},\mathsf{r_o},n)$ whose minimum distance is $d_{\min} > d^\star$. The expurgated ensemble contains at least a fraction $1-\theta > 1/2$ of the codes in the original ensemble. From [16] it is known that the average weight enumerator of the expurgated ensemble can be upper bounded by:

$$\mathcal{A}_d^{\mathrm{ex}} \begin{cases} \leq 2\mathcal{A}_d & \text{if } d > d^\star \\ = 0 & \text{if } 1 \leq d \leq d^\star \end{cases}$$

In order to obtain the simulation results of this section, in each ensemble 6000 codes[5] were selected randomly from the ensemble. For each code Monte Carlo simulations over a BEC were performed until 40 errors were collected or a maximum of 10^5 codewords were simulated. We remark that the objective here was characterizing the average performance of the ensemble, and not so much the performance of every single code (this would have required more codewords to be simulated for each code).

In Figure 6.9 we show the CER vs. the erasure probability ε for two ensembles with $\mathsf{r} = 0.9014$ and $k = 128$ that have different outer code rates, $\mathsf{r}_\mathsf{o} = 0.9275$ (good ensemble) and $\mathsf{r}_\mathsf{o} = 0.9846$ (bad ensemble). According to Figure 6.8 the good ensemble is characterized by a typical minimum distance $d^\star_{\mathrm{min}} = 2$ whereas the bad ensemble is characterized by $d^\star_{\mathrm{min}} = 0$. For both ensembles the upper bound in (6.14) holds for the average CER. However, it can be observed how the performance of the codes in the ensemble shows a high dispersion due to the short block length ($n = 142$),i.e., the CER curves of different codes from the ensemble can be quite far apart. In fact, in both ensembles a fraction of the codes has a minimum distance equal to zero, that leads to CER$= 1$ for all values of ε (around 1% for the good ensemble and 30% for the bad ensemble). If one compares Figure 6.9a and Figure 6.9b it can be seen how the fraction of codes performing close to the random coding bound is larger in the good ensemble than in the bad ensemble. For the good ensemble Figure 6.9a shows also an upper bound on the average CER for the expurgated ensemble with $d^\star = 1$. It can be observed how the expurgated ensemble has a lower error floor. For the bad ensemble no expurgated ensemble can be defined (no $d^\star \geq 0$ exists that leads to $\theta < 1/2$ in (6.15)).

Figure 6.10 shows the CER vs. ε for two ensembles using the same outer code rates as in Figure 6.9 but this time for $k = 256$. We can observe how the CER shows somewhat less dispersion compared to $k = 128$. Moreover, comparing Figure 6.10a and Figure 6.9a it can be seen how for the good ensemble ($\mathsf{r}_\mathsf{o} = 0.9275$) the error floor is much lower for $k = 256$ than for $k = 128$, due to a larger typical minimum distance. Actually, whereas for $k = 128$ there were some codes with minimum distance zero in

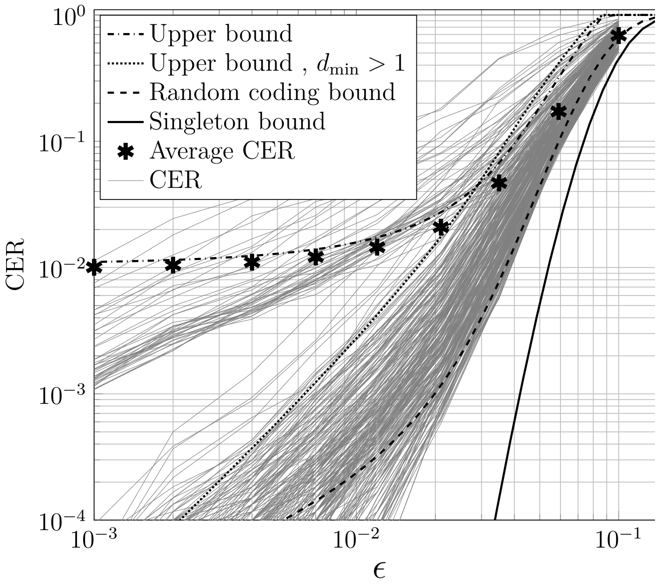
[5] For clarity of presentation only 300 codes are shown in the figures.

the good ensemble, for 256 we did not find any code with minimum distance zero out of the 6000 codes that were simulated. For the good ensemble it is possible again to considerably lower the error floor by expurgation. However, for the bad ensemble, comparing Figure 6.10b and Figure 6.9b we can see how the error floor is approximately the same for $k = 128$ and $k = 256$. In fact, in both cases the typical minimum distance is zero.

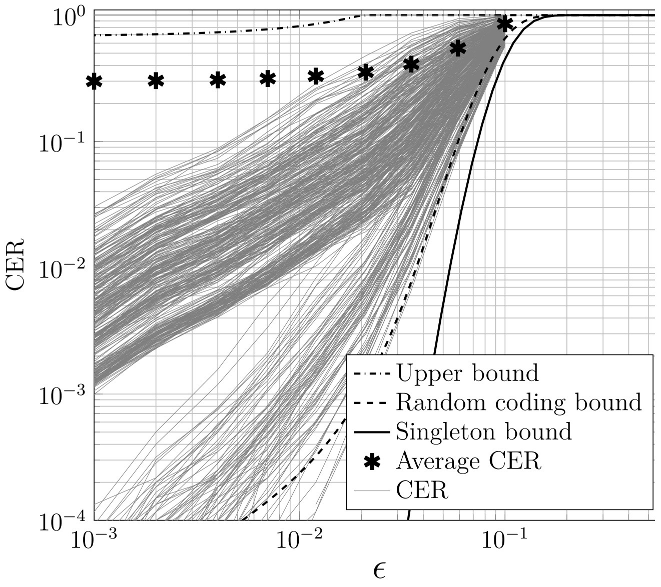
6.4.2 Comparison with Raptor Codes with Standard R10 Outer Codes

In this section we provide a numerical example that illustrates how the results obtained for linear random outer codes closely approximate the results with the standard R10 Raptor outer code (c.f. Section 3.3.2). Specifically, we consider Raptor codes with an LT degree distribution $\Omega(x) = 0.0098x + 0.4590x^2 + 0.2110x^3 + 0.1134x^4 + 0.2068x^5$. Figure 6.11 shows the positive growth rate region for such a degree distribution (assuming a linear random outer code) and three different rate points, two of which are inside the region $\mathscr{P}$ while the third one lies outside. The $(\mathsf{r_i}, \mathsf{r_o})$ rate pairs for the three points are specified in the figure caption.

In Figure 6.12 we show the average CER obtained through Monte Carlo simulations for the ensembles of Raptor codes with $k = 1024$, output degree distribution $\Omega(x)$ and two different outer codes, the standard R10 outer code and a linear random outer code. The three different rate points given in Figure 6.11 are considered. For each rate point the average CER is given for the ensemble using the standard R10 outer code and for the a linear random outer code. The CERs obtained with both precodes are very similar in all cases. Furthermore, the error floor behavior of the Raptor code ensemble with R10 outer code is in agreement with the position of the corresponding point on the $(\mathsf{r_i}, \mathsf{r_o})$ plane with respect to the $\mathscr{P}$ region, although this region is obtained using the simple linear random outer code model. Concretely, for rate points inside $\mathscr{P}$ the error floor is low, and it tends to become lower the further the point is from the boundary of $\mathscr{P}$. However, for rate points outside region $\mathscr{P}$, we have a very high error floor. Thus, our analysis, which is done for linear random outer codes, can be used to make accurate predictions on the behaviour of Raptor codes employing the R10 outer code.

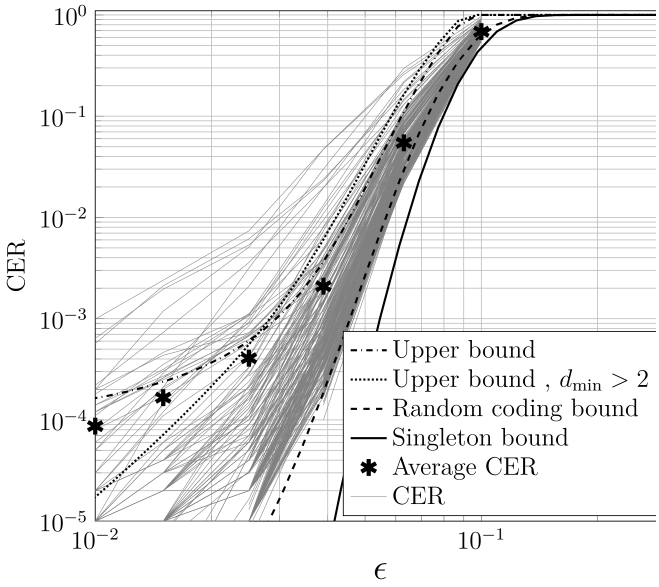


(a) good ensemble, $\mathsf{r_o} = 0.9275$, $\mathsf{r} = 0.9014$

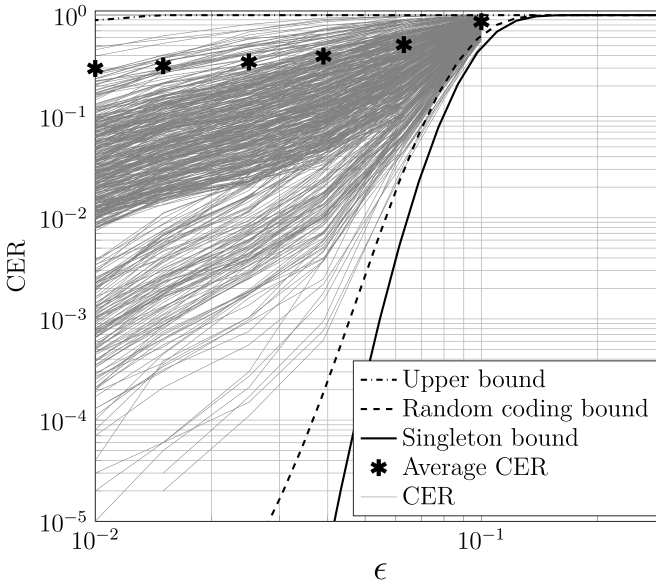


(b) bad ensemble, $\mathsf{r_o} = 0.9846$, $\mathsf{r} = 0.9014$

Fig. 6.9 Codeword error rate CER vs. erasure probability ε for two ensembles with $\mathsf{r} = 0.9014$ and $k = 128$ but different values of $\mathsf{r_o}$. The solid, dashed and dot-dashed lines represent respectively the Singleton bound, the Berlekamp random coding bound and the upper bound in (6.14). The dotted line represents the upper bound for the expurgated ensemble for $d^{\star} = 1$. The markers represent the average CER of the ensemble and the thin gray curves represent the performance of the different codes in the ensemble, both obtained through Monte Carlo simulations.



(a) good ensemble, $\mathsf{r_o} = 0.9275$, $\mathsf{r} = 0.9014$



(b) bad ensemble, $\mathsf{r_o} = 0.9846$, $\mathsf{r} = 0.9014$

Fig. 6.10 Codeword error rate CER vs. erasure probability ε for two ensembles with $\mathsf{r} = 0.9014$ and $k = 256$ but different values of $\mathsf{r_o}$. The solid, dashed and dot-dashed lines represent respectively the Singleton bound, the Berlekamp random coding bound and the upper bound in (6.14). The dotted line represents the upper bound for the expurgated ensemble for $d^\star = 2$. The markers represent the average CER of the ensemble and the thin gray curves represent the performance of the different codes in the ensemble, both obtained through Monte Carlo simulations.

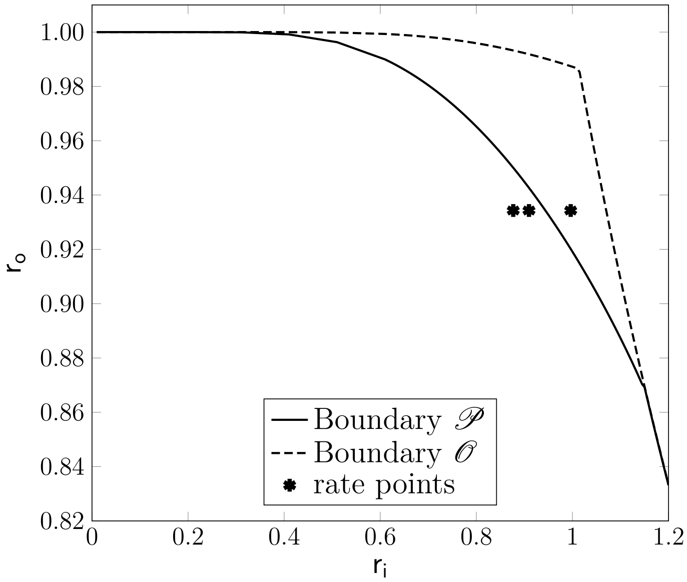


Fig. 6.11 Positive growth rate region for the degree distribution $\Omega(x) = 0.0098x + 0.4590x^2 + 0.2110x^3 + 0.1134x^4 + 0.2068x^5$. The markers represent three different rate points all of them with $\mathsf{r_o} = 1024/1096$ but with different inner code rates, $\mathsf{r_i} = 1096/1100$, $\mathsf{r_i} = 1096/1205$ and $\mathsf{r_i} = 1096/1250$.

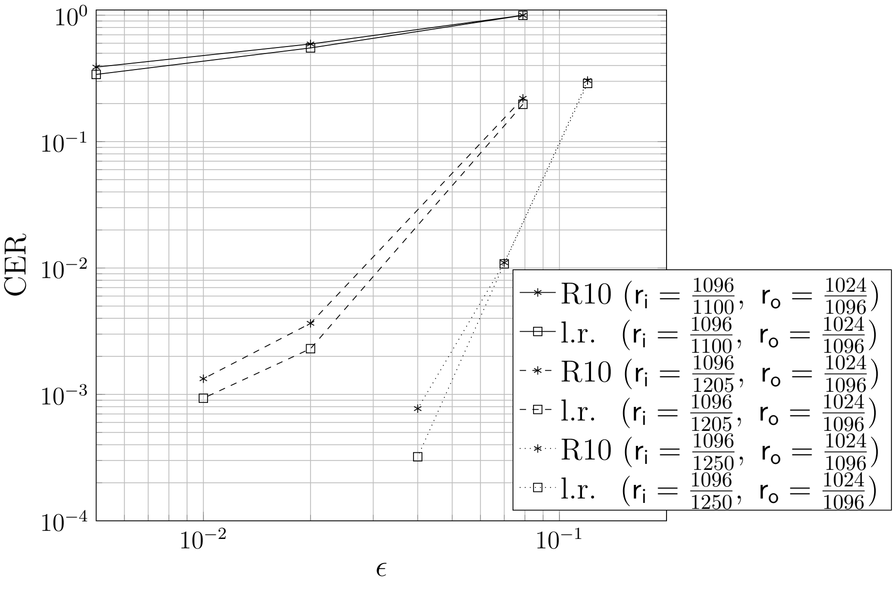


Fig. 6.12 Average CER for Raptor code ensembles using $\Omega(x) = 0.0098x + 0.4590x^2 + 0.2110x^3 + 0.1134x^4 + 0.2068x^5$ as output degree distribution and two different outer codes, the standard outer code of R10 Raptor codes and a linear random outer code, (l.r.) in the legend.

6.5 Summary

In this chapter we have analyzed the distance spectrum of fixed-rate Raptor codes with outer codes from the linear random ensemble. The expression of the average weight enumerator and the growth rate of the weight enumerator as functions of the rate of the outer code and the rate and degree distribution of the inner LT code have been derived. Based on these expressions necessary and sufficient conditions to have Raptor code ensembles with a positive typical minimum distance were derived. These conditions lead to a region $\mathscr{P}$ defined in the $(\mathsf{r_i}, \mathsf{r_o})$ plane, where $\mathsf{r_i}$ is the rate of the inner LT code and $\mathsf{r_o}$ is the rate of the outer code. Points inside region $\mathscr{P}$ correspond to fixed-rate Raptor code ensembles with a positive typical minimum distance. Moreover, a simple necessary condition has been developed too, that only requires (besides the inner and outer code rates) the knowledge of the average output degree. This condition leads to a region $\mathscr{O}$ in the $(\mathsf{r_i}, \mathsf{r_o})$ plane, which provides an outer bound to $\mathscr{P}$ and holds for all degree distributions having the same average output degree. The applicability of the theoretical results has been demonstrated by means of simulation results. Furthermore, simulation results have been presented that show that the performance of Raptor codes with linear random outer codes is close to that of Raptor codes with the standard outer code of R10 Raptor codes. Thus, we speculate that the results obtained for Raptor codes with linear random outer codes hold as first approximation for standard R10 Raptor codes.

Chapter 7

Parallel Concatenated Fountain Codes

In this chapter we present a novel fountain coding scheme that is specially suited for small values of k. The proposed scheme consists of a parallel concatenation of a (h, k) block code with a linear random fountain code (LRFC). The scheme is specially interesting when the block code is maximum distance separable (MDS). The remainder of this chapter is organized as follows. In Section 7.1 the proposed concatenated scheme is described. In Section 7.2 of the scheme is analyzed for the case in which the block code is MDS. In Section 7.3 the performance of the scheme is analyzed for a generic block code in the fixed-rate setting. In Section 7.4 numerical results are presented for a multicasting system making use of the proposed fountain coding scheme, and the performance is compared with that of LRFC codes. Finally, a summary of the results in this chapter is presented in Section 7.5.

7.1 Scheme Description

Let us define the source block $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ as a vector of source symbols belonging to a finite field of order q, i.e., $\mathbf{v} \in \mathbb{F}_q^k$. In the proposed scheme, the source block is first encoded via a (h, k) linear block code $\mathcal{C}'$ over $\mathbb{F}_q$ with generator matrix $\mathbf{G}'$. We will make use of Raptor code terminology and call this block code also precode. The encoded block is hence given by

$$\mathbf{c}' = \mathbf{v}\mathbf{G}' = (c'_1, c'_2, \ldots, c'_h).$$

Additional redundancy symbols can be obtained using an LRFC, that is, by computing linear random combinations of the k source symbols as

$$c_i = c''_{i-h} = \sum_{j=1}^{k} g_{j,i} v_j, \qquad i = h+1, \ldots, n \tag{7.1}$$

where the coefficients $g_{j,i}$ in (7.1) are selected from $\mathbb{F}_q$ uniformly at random.

Thus, the encoded sequence corresponds to:

$$\mathbf{c} = (\mathbf{c}' | \mathbf{c}'').$$

Where, $\mathbf{c}'$ and $\mathbf{c}''$ are respectively the output of the block code and the LRFC. The generator matrix of the concatenated code has the form

$$\mathbf{G} = \left(\underbrace{\begin{matrix} g_{1,1} & g_{1,2} & \cdots & g_{1,h} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,h} \end{matrix}}_{\mathbf{G}'} \middle\| \underbrace{\begin{matrix} g_{1,h+1} & g_{1,h+2} & \cdots & g_{1,n} \\ g_{2,h+1} & g_{2,h+2} & \cdots & g_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,h+1} & g_{k,h+2} & \cdots & g_{k,n} \end{matrix}}_{\mathbf{G}''} \right)$$

where $\mathbf{G}'$ and $\mathbf{G}''$ are the generator matrices of the precode and the LRFC respectively. The encoded sequence can be written as:

$$\mathbf{c} = \mathbf{v}\mathbf{G} = (c_1, c_2, \ldots, c_n).$$

This scheme can actually be seen as a parallel concatenation of the linear block code $\mathcal{C}'$ and of an LRFC (Figure 7.1), where the first h output symbols are the codeword symbols of the block code.[1]

We remark that, being the LRFC rateless, the number of output symbols n can grow indefinitely. Thus, the proposed scheme is also rateless. The encoder may be seen as a modified fountain encoder, whose first h output symbols $(c_1, c_2, \ldots, c_h)$ correspond to the codeword output by the encoder of $\mathcal{C}'$, whereas the following $n-h$ symbols are the output of the LRFC encoder. A related rateless construction was proposed in [78], where a mother non-binary LDPC code was modified by replicating the codeword symbols (prior multiplication by a non-zero field element) and thus by (arbitrarily)

[1] This represents a difference with Raptor codes, for which the output of the precode is further encoded by a LT Code. Hence the first n output symbols of a Raptor encoder do not coincide with the output of the precode.

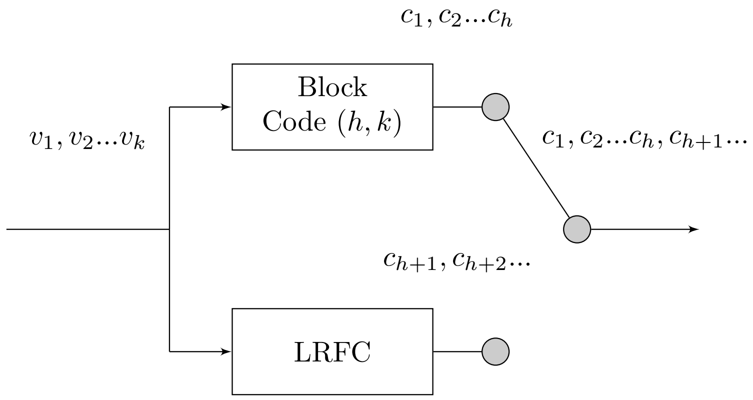


Fig. 7.1 Novel fountain coding scheme seen as a parallel concatenation of a (h, k) linear block code and a LRFC.

lowering the code rate. In our work, the mother code corresponds to the block code, and the additional redundant symbols are produced by the LRFC.

We will assume that the output symbols $\mathbf{c}$ are transmitted over an erasure channel with erasure probability ε. Let us assume that at the receiver side $m = k + \delta$ output symbols are collected, where δ is the (absolute) receiver overhead. Let us denote by $\mathscr{J} = \{j_1, j_2, \ldots, j_m\}$ the set of the indices of the output symbols of $\mathbf{c}$ that have been collected by a specific receiver. The received vector $\mathbf{y}$ is hence given by

$$\mathbf{y} = (y_1, y_2, \ldots, y_m) = (c_{j_1}, c_{j_2}, \ldots, c_{j_m})$$

and it can be related to the source block $\mathbf{v}$ as

$$\mathbf{y} = \mathbf{v}\tilde{\mathbf{G}}$$

Here, $\tilde{\mathbf{G}}$ denotes the $k \times m$ matrix made by the columns of $\mathbf{G}$ with indices in $\mathscr{J}$, i.e.,

$$\tilde{\mathbf{G}} = \begin{pmatrix} g_{1,j_1} & g_{1,j_2} & \cdots & g_{1,j_m} \\ g_{2,j_1} & g_{2,j_2} & \cdots & g_{2,j_m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,j_1} & g_{k,j_2} & \cdots & g_{k,j_m} \end{pmatrix}.$$

The recovery of $\mathbf{v}$ reduces to solving the system of $m = k + \delta$ linear equations in k unknowns

$$\tilde{\mathbf{G}}^T \mathbf{v}^T = \mathbf{y}^T. \tag{7.2}$$

The solution of (7.2) can be obtained by means of a maximum likelihood (ML) decoding algorithm (e.g., via Gaussian elimination or via inactivation decoding) if and only if $\text{rank}(\tilde{\mathbf{G}}) = k$.

7.2 Maximum Distance Separable Precode

In this section we consider the case in which the precode is maximum distance separable (MDS). The reasons to consider MDS codes are twofold. First, MDS codes meet the Singleton bound with equality, which means that over an erasure channel, decoding succeeds with probability one if the receiver is able to collect at least k symbols. Second, the use of MDS precodes leads to a very simple performance model, as it will be shown in this section. In particular, when binary codes are used, we assume $(k+1, k)$ single parity-check (SPC) codes. When operating on higher order finite fields, we consider generalized Reed-Solomon (GRS) codes.

Based on the bounds in (3.2), we will derive tight upper and lower bounds for the decoding failure probability $\mathsf{P_F}$ of our parallel concatenated fountain coding scheme for memoryless erasure channels. In our analysis we will assume that an encoded sequence $\mathbf{c}$ composed of $n \geq h$ symbols is transmitted over a memoryless erasure channel with erasure probability of ε.[2]

In our analysis we will consider two different cases. In the first case among the m received symbols, at least k have indices in $\{1, 2, \ldots, h\}$. That is, at least $m' \geq k$ symbols produced by the linear block encoder have been received. In this case, since the precode $\mathcal{C}'$ is MDS, the system of equations in (7.2) will be solvable with probability 1. The probability of this event (collecting at least k output symbols out of the first h) is given by:

$$Q(\varepsilon) = \sum_{i=k}^{h} \binom{h}{i} (1-\varepsilon)^i \varepsilon^{h-i}.$$

[2] The case $l < n$ is not considered since it is equivalent to shortening the linear block code.

In the second, less trivial case, $m' < k$ among the m received symbols have indices in $\{1, 2, \ldots, h\}$. That is, less than k output symbols from the precode are collected. This second case is complementary to the first one and will occur with probability

$$P(\varepsilon) = 1 - Q(\varepsilon).$$

In this case, matrix $\tilde{\mathbf{G}}^T$ can be partitioned as

$$\tilde{\mathbf{G}}^T = \begin{pmatrix} \tilde{\mathbf{G}}'^T \\ \tilde{\mathbf{G}}''^T \end{pmatrix} = \left(\begin{array}{cccc} g_{1,j_1} & g_{2,j_1} & \cdots & g_{k,j_1} \\ g_{1,j_2} & g_{2,j_2} & \cdots & g_{k,j_2} \\ \vdots & \vdots & \ddots & \vdots \\ g_{1,j_{m'}} & g_{2,j_{m'}} & \cdots & g_{k,j_{m'}} \\ \hline g_{1,j_{m'+1}} & g_{2,j_{m'+1}} & \cdots & g_{k,j_{m'+1}} \\ g_{1,j_{m'+2}} & g_{2,j_{m'+2}} & \cdots & g_{k,j_{m'+2}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{1,j_m} & g_{2,j_m} & \cdots & g_{k,j_m} \end{array} \right). \tag{7.3}$$

The fact that the precode $\mathcal{C}'$ is MDS assures that $\text{rank}(\tilde{\mathbf{G}}') = m'$, i.e., the first m' rows of $\tilde{\mathbf{G}}^T$ are linearly independent. The remaining rows of $\tilde{\mathbf{G}}^T$ correspond to $\tilde{\mathbf{G}}''^T$ that has size $m'' \times k$, with $m'' = m - m'$. The elements in $\tilde{\mathbf{G}}''^T$ are uniformly distributed in $\mathbb{F}_q$. It follows that the matrix in (7.3) can be put (via column permutations over $\tilde{\mathbf{G}}^T$ and row permutations/combinations over $\tilde{\mathbf{G}}'^T$) in the form

$$\hat{\mathbf{G}}^T = \left(\begin{array}{c|c} \mathbf{I} & \mathbf{A} \\ \hline \mathbf{0} & \mathbf{B} \end{array} \right),$$

where $\mathbf{I}$ is the $m' \times m'$ identity matrix, $\mathbf{0}$ is a $m'' \times m'$ all-0 matrix, and $\mathbf{A}$, $\mathbf{B}$ have respective sizes $m' \times (k - m')$ and $m'' \times (k - m')$. The lower part of $\hat{\mathbf{G}}^T$, given by $(\mathbf{0}|\mathbf{B})$, is obtained by adding to each row of $\tilde{\mathbf{G}}''^T$ a linear combination of rows from $\tilde{\mathbf{G}}'^T$, in a way that the m' leftmost columns of $\tilde{\mathbf{G}}''^T$ are all set to zero. Thus, the elements of submatrix $\mathbf{B}$ are obtained by adding a deterministic symbol of $\mathbb{F}_q$ to an element of $\tilde{\mathbf{G}}''^T$, which is uniformly distributed in $\mathbb{F}_q$. It follows that the statistical properties of $\tilde{\mathbf{G}}''^T$ are inherited by the $m'' \times (k - m')$ submatrix $\mathbf{B}$, whose elements are, hence, uniformly distributed in $\mathbb{F}_q$. It follows that (7.2) is solvable if and only if $\mathbf{B}$ is full rank, i.e., if and only if $\text{rank}(\mathbf{B}) = k - m'$. Let us denote the decoding failure event as F.

The conditional decoding failure probability can be expressed as

$$\Pr\{F|m', m' < k, \delta\} = \Pr\{\text{rank}(\mathbf{B}) < k - m'\}. \tag{7.4}$$

Matrix $\mathbf{B}$ is a $m'' \times (k - m') = (k + \delta - m') \times (k - m')$ random matrix having δ rows in excess with respect to the number of columns. Hence, we can replace (7.4) in (3.2), obtaining the bounds

$$q^{-\delta-1} \leq \Pr\{F|m', m' < k, \delta\} < \frac{1}{q-1} q^{-\delta}. \tag{7.5}$$

The bounds in (3.2) are independent from the size of the matrix, they depend only on the overhead. Therefore, we can remove the conditioning on m' from (7.5), leaving

$$q^{-\delta-1} \leq \Pr\{F|m' < k, \delta\} < \frac{1}{q-1} q^{-\delta}.$$

The failure probability can now be expressed as

$$\begin{aligned} \mathsf{P_F} = \quad & \Pr\{F|m' < k, \delta\} \Pr\{m' < k\} \\ & + \Pr\{F|m' \geq k, \delta\} \Pr\{m' \geq k\} \end{aligned}$$

where $\Pr\{F|m' \geq k, \delta\} = 0$ (since at least k of the symbols produced by the MDS encoder have been collected) and $\Pr\{m' < k\} = P(\varepsilon)$. It results that

$$P(\varepsilon) q^{-\delta-1} \leq \mathsf{P_F} < P(\varepsilon) \frac{1}{q-1} q^{-\delta}. \tag{7.6}$$

If one inspects (3.2) and (7.6), one can see how the bounds on the failure probability of the concatenated scheme are scaled down by a factor $P(\varepsilon)$, which is a monotonically increasing function of ε. Therefore, when the channel conditions are *bad* (i.e., for large ε) $P(\varepsilon) \to 1$, and the bounds in (7.6) tend to coincide with the bounds in (3.2). On the other hand, if the channel conditions are *good* (i.e., for small ε), most of the time $m' \geq k$ symbols produced by the linear block encoder are received and decoding succeeds (recall the assumption of MDS code). In these conditions, $P(\varepsilon) \ll 1$, and according to the bounds in (7.6) the failure probability may decrease by several orders of magnitude.

Given the fact that the probability of decoding failure of the concatenated scheme is a function of the erasure probability, the scheme is not universal anymore in a strict

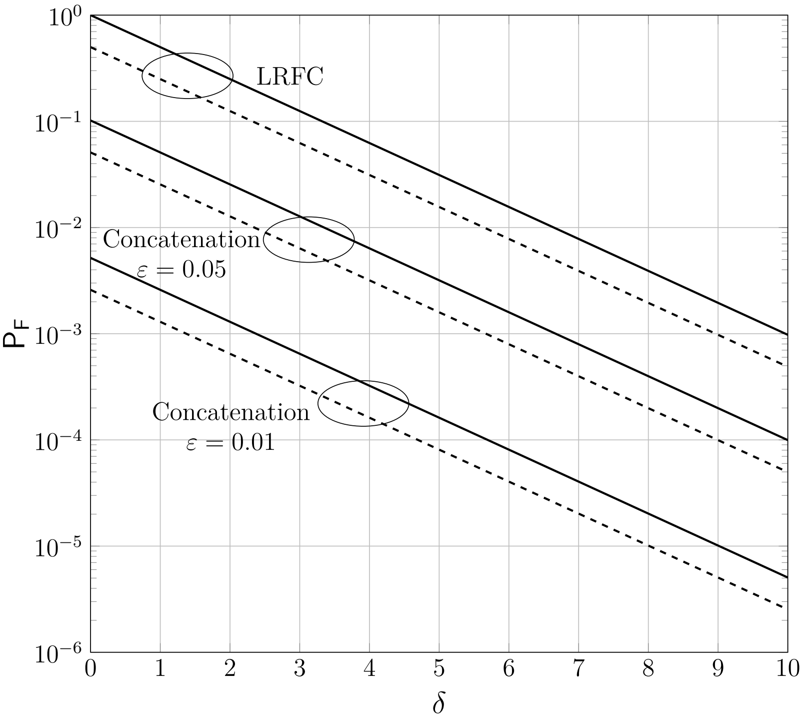


Fig. 7.2 $\mathsf{P_F}$ vs. δ for a concatenated code built using a $(11, 10)$ SPC code over $\mathbb{F}_2$ for different values of erasure probability ε. Upper bounds are represented by solid lines and lower bounds are represented by dashed lines.

sense[3]. At low channel erasure probabilities the proposed scheme will outperform LRFCs, whereas for large erasure probabilities it will perform as LRFCs. Hence, the performance of our scheme is lower bounded by that of LRFC, which are universal codes (their performance depends only on the number of output symbols received and not on the erasure probability of the channel). Therefore, one could argue that the proposed scheme is universal in a broad sense, although its probability of decoding failure does depend on the erasure probability of the channel

Figure 7.2 shows the probability of decoding failure $\mathsf{P_F}$ as a function of the number of overhead symbols δ for a concatenated code built using a $(11, 10)$ SPC code over $\mathbb{F}_2$. We can observe how, for lower erasure probabilities, the gain in performance of the concatenated code with respect to a LRFC is larger. For $\varepsilon = 0.01$ the decoding failure probability is more than 2 orders of magnitude lower than that of a LRFC.

[3] In this concatenated fountain coding scheme the output symbols are not statistically identical and independent from each other. As a consequence its performance depends on the channel erasure rate and its performance will also vary if the channel is not memoryless.

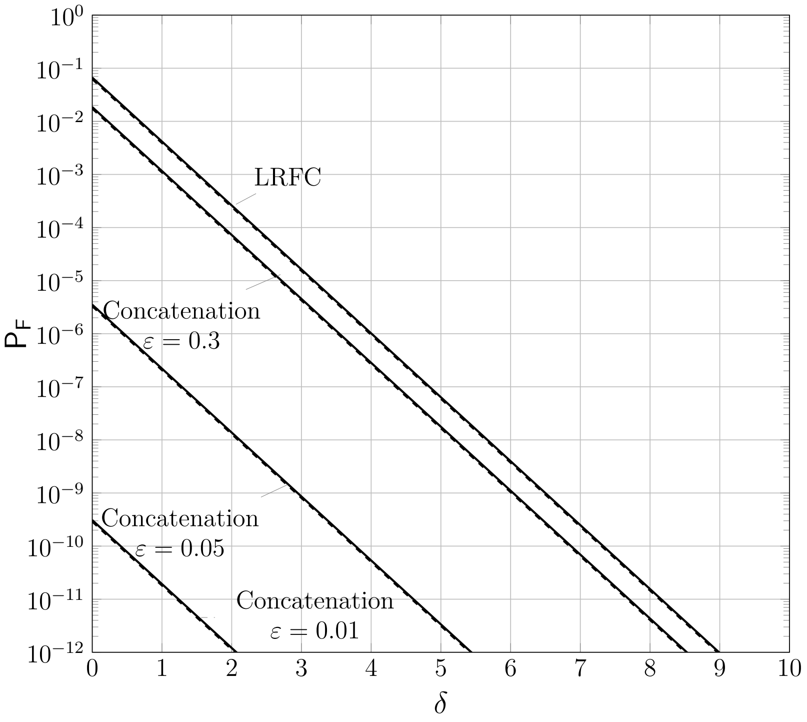


Fig. 7.3 $\mathsf{P_F}$ vs. δ for a concatenated code built using a $(15, 10)$ RS over $\mathbb{F}_{16}$ for different values of of erasure probability ε. Upper bounds are represented by solid lines and lower bounds are represented by dashed lines.

Figure 7.3 shows the probability of decoding failure vs. the number of overhead symbols δ for the concatenation of a $(15, 10)$ RS and a LRFC over $\mathbb{F}_{16}$. The performance of the concatenated code is compared with that of the LRFC built on the same field for different erasure probabilities. In this case the decrease in terms of probability of decoding failure is even more notable than in binary case. For a channel with an erasure probability $\varepsilon = 0.05$, the probability of decoding failure of the concatenated scheme is 4 orders of magnitude lower than that of the LRFC. If we compare Figure 7.3 with Figure 7.2, we can observe how the upper and lower bounds are closer to each other for the codes constructed over $\mathbb{F}_{16}$ compared to the binary codes. This effect stems from the fact that the bounds in (3.2) become tighter as the Galois field order q increases.

Figure 7.4 shows the probability of decoding failure $\mathsf{P_F}$, as a function of the receiver overhead δ, obtained via Monte Carlo simulations together with the bounds in (7.6). The results refer to a concatenation of a $(11, 10)$ SPC with an LRFC over $\mathbb{F}_2$, and a

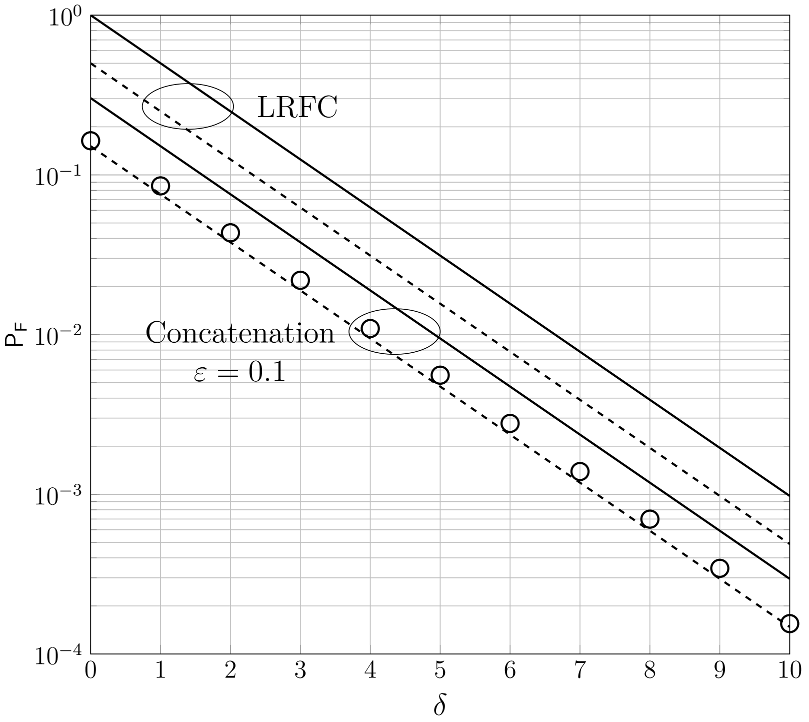


Fig. 7.4 $\mathsf{P_F}$ vs. δ for the concatenation of a $(11, 10)$ SPC code and a LRFC over $\mathbb{F}_2$ and $\varepsilon = 0.1$. Upper bounds are represented by solid lines and lower bounds are represented by dashed lines. The points marked with '$\circ$' denote actual simulations.

channel with an erasure probability $\varepsilon = 0.1$. As expected, the simulation results tightly match the bounds.

Figure 7.5 shows similar simulation results for a concatenation of a $(15, 10)$ RS code with an LRFC over $\mathbb{F}_{16}$, for a channel erasure probability $\varepsilon = 0.1$. Also in this case, the results are very close to the bounds.

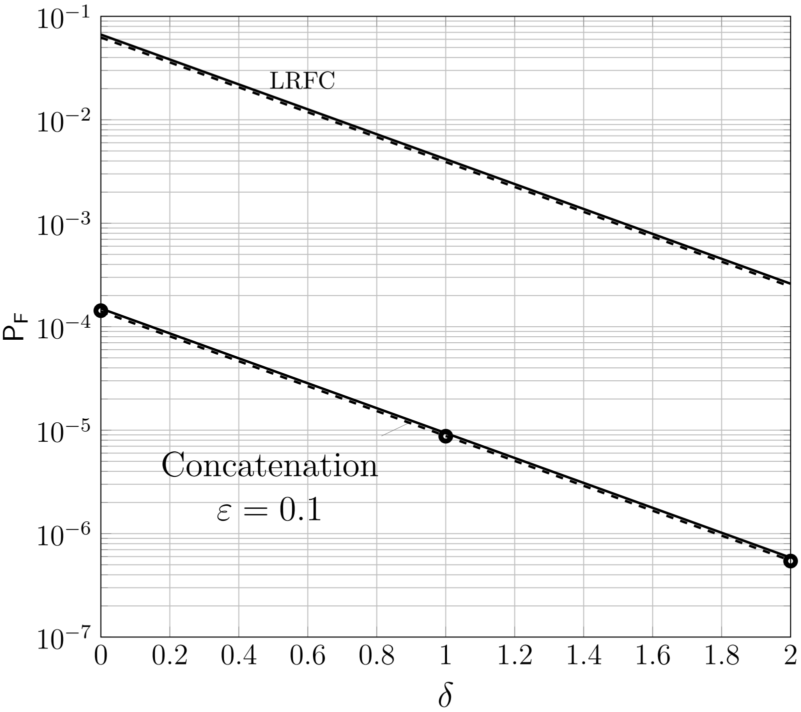


Fig. 7.5 $\mathsf{P_F}$ vs. δ for the concatenation of a $(15, 10)$ RS and LRFC over $\mathbb{F}_{16}$ and $\varepsilon = 0.1$. Upper and lower bounds are represented by solid and dashed lines, respectively. The markers '$\circ$' denote simulations.

7.3 Generic Precode in a Fixed-Rate Setting

Fountain codes are often used in a fixed-rate setting (see Chapter 6). In this context, the main advantage in the use of fountain codes with respect to block erasure correcting codes stems from the possibility of adapting code rate and block length to the transmission needs (e.g., channel conditions) in a flexible manner. In this section, we consider the concatenated scheme in the general case where the block code $\mathcal{C}'$ is not necessarily maximum distance separable (MDS) in a fixed-rate setting. We derive the weight enumerator (WE) of the concatenated code and use it to derive a tight upper bound on the block error probability of the code.

The coding scheme considered in this chapter is a parallel concatenation of a linear block code and a LRFC, which for a finite rate setting is a random generator matrix code. Let us denote as $\mathscr{C}(\mathcal{C}', k, n, q)$ the ensemble of codes obtained by a parallel-concatenation of a (h, k) linear block code over $\mathbb{F}_q$, $\mathcal{C}'$, with all possible realizations of an LRFC, where k is the number of source symbols, n is the total number of output symbols and q is the finite field order. Note that the codes in the ensemble have fixed-rate $r = k/n$. We denote as $\mathscr{A}_i(X)$ the conditional output-weight enumerator function (CO-WEF) averaged over the ensemble $\mathscr{C}(\mathcal{C}', k, n, q)$ conditioned to the input source block having weight i,

$$\mathscr{A}_i(X) = \sum_{w=1}^{n} \mathcal{A}_{i,w} X^w$$

where $\mathcal{A}_{i,w}$ is the average number of codewords of Hamming weight w produced by Hamming weight-i inputs, that is, $\mathcal{A}_{i,w}$ is the input output-weight enumerator function (IO-WEF) of the code. For the ensemble of parallel-concatenated codes the average conditional output-weight enumerator function admits a very compact expression:

$$\mathscr{A}_i(X) = \frac{\mathscr{A}_i^{\mathcal{C}'}(X)\mathscr{A}_i^{\mathscr{L}(k,l,q)}(X)}{\binom{k}{i}}, \tag{7.7}$$

where $\mathscr{A}_i^{\mathcal{C}'}(X)$ is the conditional output-weight enumerator function of the linear block code, and $\mathscr{A}_i^{\mathscr{L}(k,l,q)}(X)$ is the average conditional output-weight enumerator function of the ensemble $\mathscr{L}(k, l, q)$, being $\mathscr{L}(k, l, q)$ the ensemble of linear block codes over $\mathbb{F}_q$

with $k \times l$ generator matrix $\mathbf{G}''$, with $l = n - h$. Let us assume that $\mathscr{A}_i^{\mathcal{C}'}(X)$ is known.[4] In this case, the derivation of $\mathscr{A}_{i,w}$ reduces to the calculation of $\mathscr{A}_i^{\mathscr{L}(k,l,q)}(X)$.

The average number of codewords of Hamming weight w produced by Hamming weight-i inputs for the ensemble $\mathscr{L}(k,l,q)$, $\mathcal{A}_{i,w}^{\mathscr{L}(k,l,q)}$, is given by:

$$\mathcal{A}_{i,w}^{\mathscr{L}(k,l,q)} = \binom{k}{i}\binom{l}{w} p_i^w \left(1 - p_i\right)^{l-w}, \tag{7.8}$$

where p_i the probability of one of the l output symbols having a non-zero value conditioned to having an input of Hamming weight i. Given that the coefficients of $\mathbf{G}''$ are picked with uniform probability over $\mathbb{F}_q$, we have that[5]

$$\begin{array}{ll} p_i = \frac{q-1}{q} & , \; i \neq 0 \\ p_i = 0 & , \; i = 0. \end{array} \tag{7.9}$$

Using (7.7), (7.8) and (7.9) the conditional output-weight enumerator function of our concatenated scheme is obtained.

Once the conditional output-weight enumerator function has been derived, the average weight enumerator function (WEF) $\mathscr{A}(X)$ can be obtained from the average conditional output-weight enumerator function by summing over all possible input weights

$$\mathscr{A}(X) = \sum_i \mathscr{A}_i X^i.$$

Finally, the average number of codewords of Hamming weight w (average weight enumerator) $\mathcal{A}_w$ is simply obtained as the coefficient of degree w in the WEF. The average weight enumerator of the concatenated ensemble can be used now to derive a tight upper bound on the expected block error probability for the codes of the ensemble using Di's upper bound, (2.3) [46].

As an example, we consider a concatenated scheme where the block code is a binary $(63, 57)$ Hamming code. The conditional output-weight enumerator function $\mathscr{A}_i(X)$ of

[4] In general, the derivation of the conditional output-weight enumerator function $\mathscr{A}_i^{\mathcal{C}'}(X)$ for a code is not trivial, unless the code $\mathcal{C}'$ (or its dual code) has small dimension [73].

[5] Note that when $i = 0$ the encoder input is given by the all-zero word. Thus, the encoder output is zero with probability 1 due to the linearity of the code ensemble $\mathscr{L}(k,l,q)$.

a $(h = 2^t - 1, k = h - t)$ Hamming code is known from [79] and corresponds to

$$\begin{aligned}\mathscr{A}(x,X) =& \frac{(1+x)^{2^{t-1}-t-1}}{2^t} \times \Big(2^t(1-x)^{2^{t-1}-t}(1-xX)^t \\ & - (1-x)^{2^{t-1}}(1+X)^t + (1+x)^{2^{t-1}}(1+X)^t\Big)\end{aligned}$$

where $\mathscr{A}(x,X) = \sum_i \mathscr{A}_i(X)x^i$.

Figure 7.6 shows the average weight enumerator vs. the normalized weight, $\varpi = w/n$ for the concatenated code for rates $r = 1/2$ and $r = 1/4$ and the weight enumerator of the precode alone (Hamming). The figure also shows the average weight enumerator of binary linear random generator matrix based ensembles with the same block length and rate. Codes in this ensemble are characterized by having a $k \times n$ generator matrix whose elements are picked uniformly at random in the binary field. Thus the ensemble is equivalent to the fixed-rate LRFC ensemble. The average weight enumerator of this ensemble can be found in [80]. In the figure we can observe how the weight spectrum of the concatenated ensemble is better than that of the binary linear random ensemble, in the sense that for same block length and rate the expected multiplicity of low weight codewords is lower. This will lead to a lower error floor.

Figure 7.7 shows the upper bounds on the codeword error rate (CER) of the ensemble, as a function of the channel erasure probability ε for different coding rates. The solid lines represent the upper bound on the CER in (2.3), and the dashed and red lines represent respectively the Berlekamp random coding bound [44], which is an upper bound on the average block error probability of random codes, and the Singleton bound, which provides the block error probability of MDS codes. The markers represent the results of Monte Carlo simulations. In order to obtain average results for the ensemble, the CER was averaged over 1000 different LRFC realizations. As expected, the bound in (2.3) is very tight in all cases. Results for three different rates are shown in the figure. The highest rate corresponds to the use of the Hamming code alone, and the other two rates are $r = 0.8$ and $r = 0.5$. While for the Hamming code the performance lies in between the one of random codes and the one of MDS codes, as the code rate decreases the performance of the scheme gets closer to the Berlekamp random coding bound, which means that for low rates our scheme performs almost as a random code. However, for high rates the concatenated scheme performs substantially better than a random code, whose performance would be very close to the Berlekamp bound.

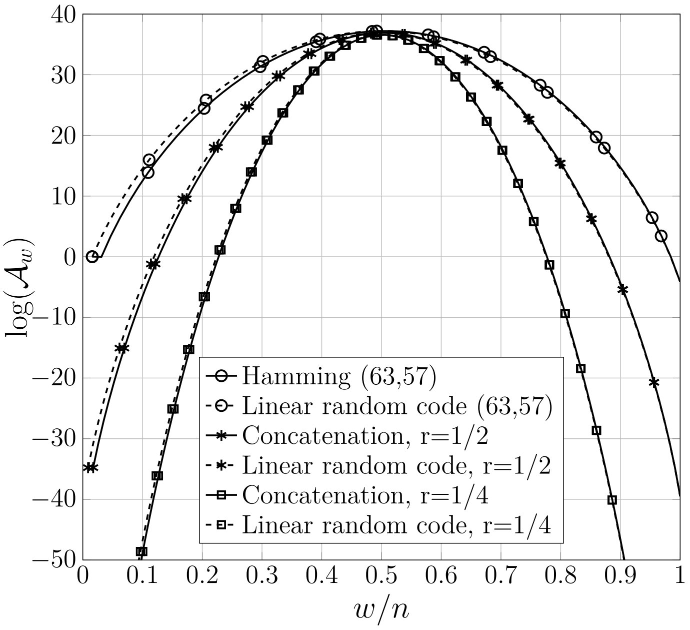


Fig. 7.6 $\log(\mathcal{A}_w)$ vs. w/n for the concatenation of a (63,57) Hamming code with a LRFC code in $\mathbb{F}_2$ and for the concatenated scheme with rates $r = \frac{1}{2}$ and $r = \frac{1}{4}$.

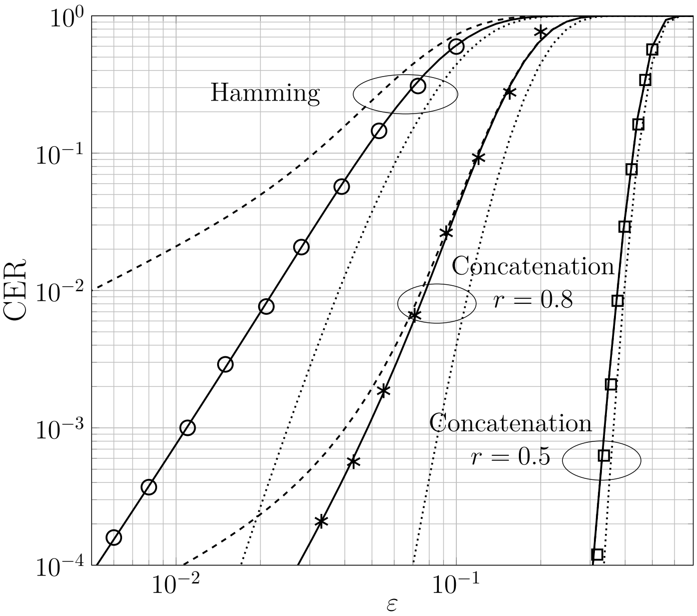


Fig. 7.7 CER vs. erasure probability ε for the concatenation of a (63,57) Hamming code with a LRFC code in $\mathbb{F}_2$. The markers represent the result of Monte Carlo simulations. The solid line represents the upper bound in [46], and the dashed and dotted lines represent the Berlekamp random coding bound and the Singleton bound respectively.

7.4 Numerical Results

In this section we investigate the performance of the concatenated scheme in a reliable multicasting scheme. Let us assume a transmitter wants to deliver a source block (a data file) to a set of N of receivers. We will assume that the erasure channels from the transmitter to the different receivers are independent and have an identical erasure probability ε. Furthermore, we assume that the receivers send an acknowledgement to the transmitter whenever they successfully decode the source block though an ideal (error- and delay-free) feedback channel. After retrieving all the acknowledgments, the transmitter stops encoding additional symbols from the source block.

Let us denote by Δ the number of symbols in excess with respect to k transmitted by the sender. We refer to Δ as the transmission overhead. When $k+\Delta$ symbols have been transmitted, the probability that a specific receiver gathers exactly m symbols is

$$S\left(\Delta, m\right) = \binom{k+\Delta}{m} (1-\varepsilon)^m \varepsilon^{k+\Delta-m}.$$

The probability of decoding failure at the receiver given that the transmitter has sent $k+\Delta$ symbols is hence

$$\begin{aligned} \mathsf{P_e} = &\sum_{m=0}^{k-1} S\left(\Delta, m\right) + \\ &+ \sum_{m=k}^{k+\Delta} S\left(\Delta, m\right) \mathsf{P_F} | (\delta = m-k, \varepsilon). \end{aligned}$$

Let us define the error probability in our system, $\mathsf{P_E}$, as he probability that at least one receiver is not able to decode the source block. This probability is given by

$$\mathsf{P_E}(N, \Delta, \varepsilon) = 1 - (1 - \mathsf{P_e})^N$$

Observe that $\mathsf{P_E}(N, \Delta, \varepsilon)$ can be easily bounded by means of (7.6). Following this approach, we compare the performance of the proposed concatenation to that of LRFCs and to that of an ideal fountain code. Let us recall that for an ideal fountain code the probability of decoding failure is zero whenever k or more output symbols are collected (see Section 2.3).

We consider a system with $N = 10^4$ receivers and a channel with an erasure probability $\varepsilon = 0.01$. The performance of LRFC codes over $\mathbb{F}_2$ and $\mathbb{F}_{16}$ is depicted in Figure 7.8 together with that of two concatenated schemes: a concatenation of a

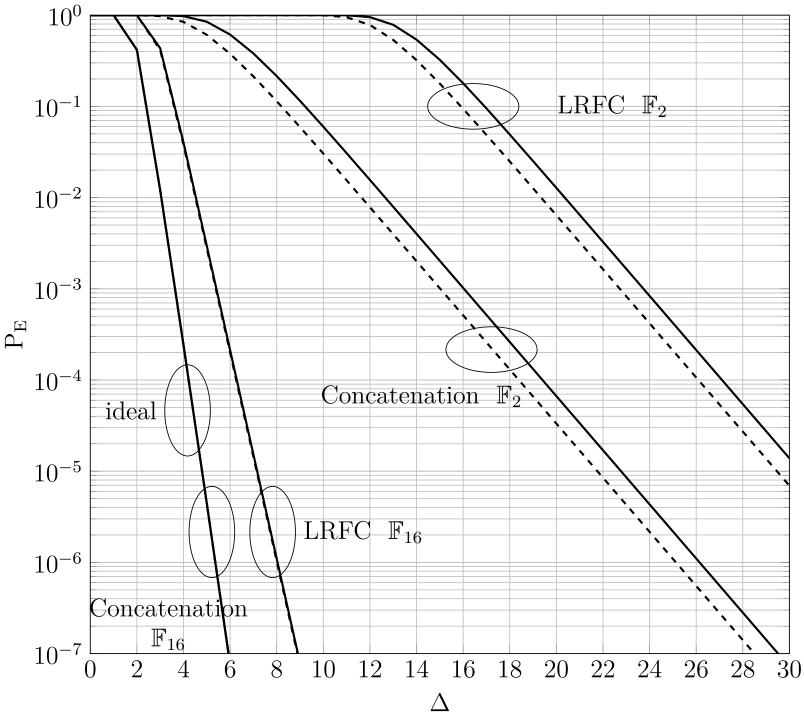


Fig. 7.8 $\mathsf{P_E}$ vs. overhead at the transmitter in a system with $N = 10000$ receivers and $\varepsilon = 0.01$. Results are shown for different fountain codes: LRFC in $\mathbb{F}_2$, LRFC in $\mathbb{F}_{16}$, concatenation of a (11,10) SPC code with a LRFC code in $\mathbb{F}_2$, and a concatenation of a $(15, 10)$ RS code and a LRFC code over $\mathbb{F}_{16}$.

$(11, 10)$ SPC code with a LRFC code over $\mathbb{F}_2$, and a concatenation of a $(15, 10)$ RS code and a LRFC code over $\mathbb{F}_{16}$. We can observe how the binary concatenated scheme outperforms the binary LRFC. For example, in order to achieve a target probability of error $\mathsf{P_E} = 10^{-4}$ the concatenated scheme needs only $\Delta = 20$ overhead symbols whereas the LRFC requires a transmission overhead $\Delta = 27$. In the binary case, both the LRFC and the concatenated scheme are far from the performance of an ideal fountain code. If we now look at the non binary case, we can observe how the performance gap of the LRFC w.r.t an ideal fountain code is much smaller than in the binary case. Furthermore, we can observe how the non-binary concatenated scheme is able to improve the performance of the LRFC and almost completely close the performance gap w.r.t. an ideal fountain code.

7.5 Summary

In this chapter a novel coding scheme has been introduced. The scheme consists of a parallel concatenation of a block code with a linear random fountain code (LRFC) code, both constructed over the same finite field. The performance of the concatenated coding scheme under ML decoding has been analyzed through the derivation of tight bounds on the probability of decoding failure. This scheme can be seen as a way of turning any block code rateless (or rate flexible), so that additional output symbols can be generated on demand. The proposed scheme is in general only practical when the code dimension k is small.

Specially interesting is the case in which the block code is a MDS code. In this case, the scheme can provide failure probabilities lower than those of LRFC codes by several orders of magnitude, provided that the erasure probabilities of the channel is not too high. The general case in which the block code is not MDS has also been analyzed. In this case the scheme has been analyzed in a fixed-rate setting, and it has been shown by means of examples how the concatenated scheme outperforms LRFCs. Given the fact that Raptor codes are essentially random codes, their performance would be at best as good as that of LRFCs. Thus, the proposed scheme also outperforms Raptor codes in terms of decoding failure probability. However, one should remark that the proposed scheme is only practical when the code dimension k is small, since its decoding complexity is rather high.

The focus in this chapter has been exclusively on the performance under ML decoding. However, it is possible to exploit the structure of the block code (precode) in order to decrease the decoding complexity of the scheme. For example, in [37] an enhanced decoding algorithm was proposed for the case in which the precode is a generalized Reed-Solomon (GRS) code whose generator matrix is in Vandermonde form.

Chapter 8

Conclusion

In this dissertation we have investigated fountain codes under maximum likelihood (ML) erasure decoding. In particular three types of fountain codes have been considered, LT codes, Raptor codes and a new class of parallel concatenated fountain codes.

Regarding LT codes, the main contribution of this thesis is a detailed analysis of a particular ML decoding algorithm, inactivation decoding. More concretely, the focus has been on the decoding complexity of LT codes under inactivation decoding in terms of the number of inactivations. Given an LT degree distribution and k, the code dimension or equivalently the number of input symbols, dynamic programming approaches have been used to derive the expected number of inactivations and its probability distribution. Furthermore, a low complexity algorithm has been proposed to estimate the number of inactivations. Additionally, we have shown by means of an example how the analysis of LT codes presented can be used to numerically design LT codes optimized for inactivation decoding.

Raptor codes have also been considered in this thesis. First, upper bounds to the probability of decoding failure of Raptor codes have been derived, using the weight enumerator of the outer code, or the average weight enumerator when the outer code is drawn at random from an ensemble. These bounds show that Raptor codes can be analyzed similarly to fixed-rate block codes. Furthermore, we have shown how the complexity of Raptor codes under inactivation decoding can be approximated introducing the concept of a surrogate LT code. Moreover, we have shown by means of an example how the results obtained for Raptor codes can be used to design finite length Raptor codes with a good tradeoff between probability of decoding failure and complexity under inactivation decoding.

Additionally, an analysis of the distance spectrum of ensembles of fixed-rate Raptor codes has been presented, for the case in which the outer code is picked from the linear random ensemble. This ensemble of Raptor codes resembles standard R10 Raptor codes as a first order approximation. For this ensemble, the average weight enumerator and its growth rate have been derived. Furthermore, sufficient and necessary conditions for the ensemble to have a minimum distance growing linearly with the block length (positive typical minimum distance) have been derived. By means of simulations, it has been shown how the results obtained for the ensemble of Raptor codes studied can be extrapolated to Raptor codes using the standard R10 outer code as a first approximation.

The last contribution of the dissertation is the introduction of a novel class of fountain codes, that consists of a parallel concatenation of a block code with a linear random fountain code (LRFC). This scheme is specially interesting when the block code is a maximum distance separable (MDS) code. In this case, the scheme's performance can be tightly upper and lower bounded by means of very simple formulae. Furthermore, the scheme can provide failure probabilities lower than those of LRFC codes by several orders of magnitude, provided that the erasure probabilities of the channel is not too high, which is usually the case in most of the applications of erasure codes. Thus, in this setting the proposed scheme outperforms Raptor codes in terms of probability of decoding failure. However, this novel scheme is in general only practical for when the code dimension k is small due to its high decoding complexity.

Appendix A

Comparison of Inactivation Strategies

In this appendix the performance of the different inactivation techniques presented in Section 4.1.1 are compared by means of simulations. More concretely, we simulated a (non-systematic) R10 Raptor code for source block sizes ranging from $k = 128$ to $k = 8192$ for different absolute overheads δ. For each different value of δ, 300 decodings were carried out and the average number of inactivations was obtained for random inactivation, maximum reduced degree inactivation, maximum accumulated degree inactivation and maximum component inactivation.

The simulation results can be observed in Figures A.1 to A.7. Looking at these figures it can be observed how random inactivation leads to the largest number of inactivations, followed by maximum reduced degree inactivation, then maximum accumulated degree and finally maximum component inactivation, that leads to the least inactivations (lowest decoding complexity). It is remarkable that this ordering holds for all values of k and δ.

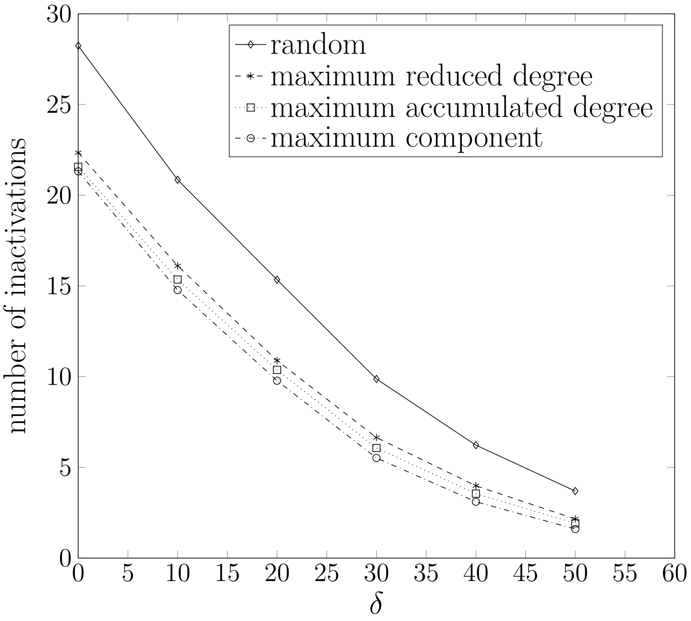


Fig. A.1 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 128$

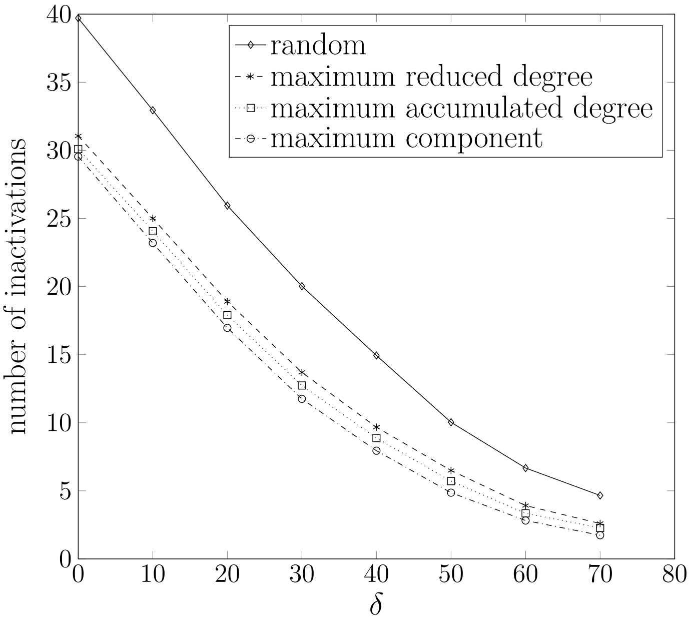


Fig. A.2 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 256$

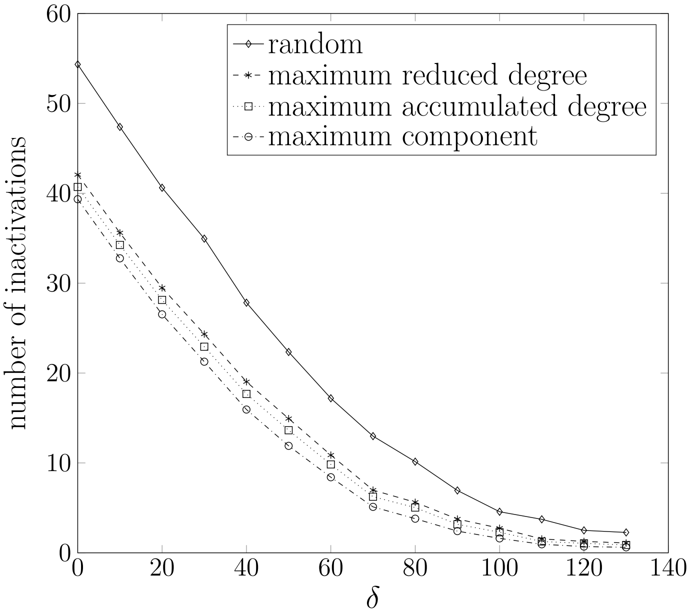


Fig. A.3 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 512$

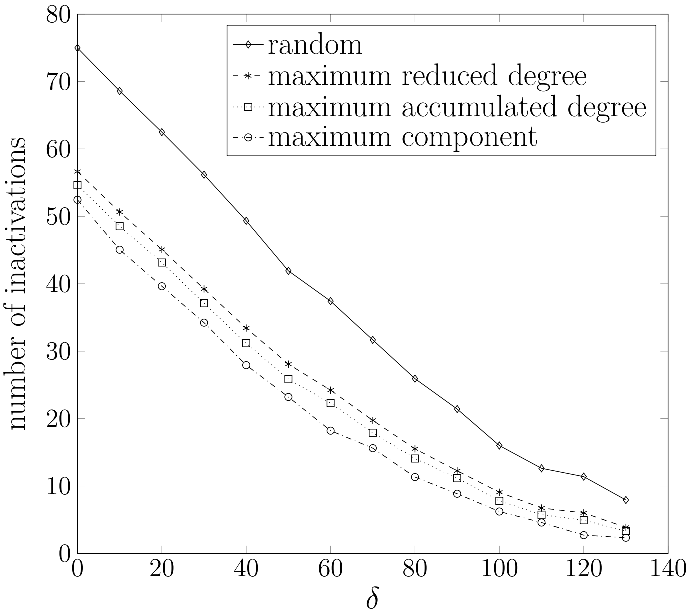


Fig. A.4 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 1024$

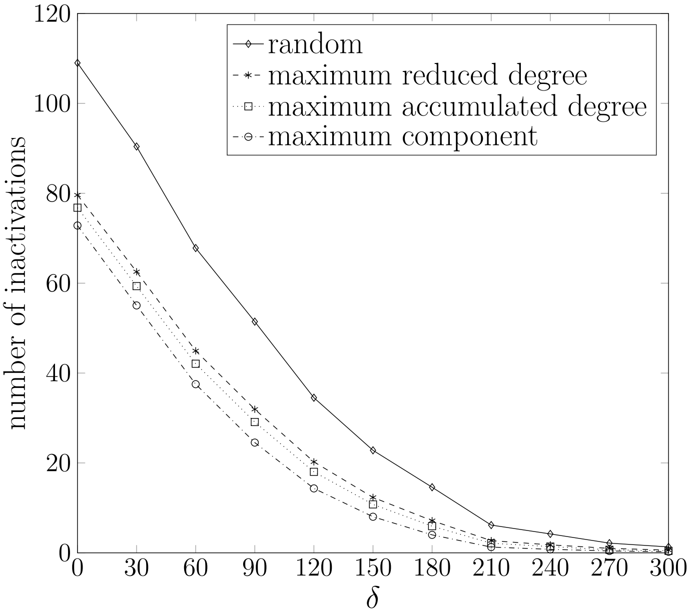


Fig. A.5 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 2048$

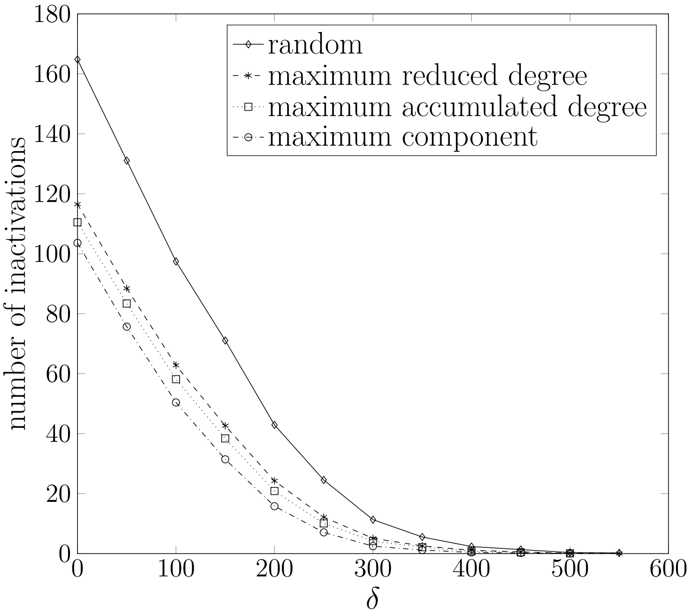


Fig. A.6 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 4096$

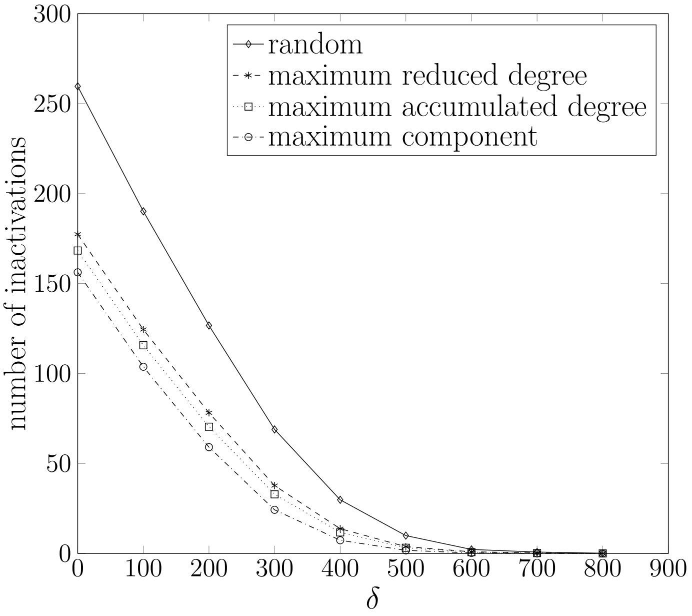


Fig. A.7 Number of inactivations vs. absolute receiver overhead δ for a R10 Raptor code and $k = 8192$

Appendix B

Omitted Proofs

In this Appendix we provide some results that were omitted in the body of the thesis but are necessary for the proofs in Chapters 5 and 6.

B.1 Proof of Theorem 1

The following lemma is used in the proof of Theorem 1.

Lemma 3. *Let X_1, X_2 ... X_l be i.i.d random variables with uniform distribution over $\mathbb{F}_{2^m}\backslash\{0\}$. Then*

$$\Pr\{X_1 + X_2 + \ldots + X_l = 0\} = \frac{1}{q}\left(1 + \frac{(-1)^i}{(q-1)^{i-1}}\right).$$

Proof. The proof starts observing that the additive group of $\mathbb{F}_{2^m}$ is isomorphic to the vector space $\mathbb{Z}_2^m$. Thus, we consider X_1, X_2 ... X_l to be i.i.d random variables with uniform distribution over the vector space $\mathbb{Z}_2^m\backslash\{0\}$.

Let us introduce the auxiliary random variable

$$W := X_1 + X_2 + \ldots + X_l$$

Denote by $P_W(w)$ and by $P_X(x)$ the probability mass function of W and X_i, with

$$P_X(x) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{1}{q-1} & \text{otherwise.} \end{cases}$$

We have that

$$P_W(w) = P_X(x) * P_X(x) * \ldots * P_X(x)$$

which can be re-stated via the m-dimensions 2-points DFT $\mathscr{I}\{\cdot\}$ as

$$\mathscr{I}\{P_W(w)\} = \left(\mathscr{I}\{P_X(x)\}\right)^l.$$

We have that

$$\hat{P}_X(t) := \mathscr{I}\{P_X(x)\} = \begin{cases} 1 & \text{if } t = 0 \\ \frac{-1}{q-1} & \text{otherwise} \end{cases}$$

Thus,

$$\hat{P}_W(t) := \mathscr{I}\{P_W(w)\} = \begin{cases} 1 & \text{if } t = 0 \\ \frac{(-1)^l}{(q-1)^l} & \text{otherwise.} \end{cases}$$

We are interested in $P_W(0)$ whose expression corresponds to

$$P_W(0) = \frac{1}{q}\sum_t \hat{P}_W(t) = \frac{1}{q} + \frac{1}{q}(q-1)\frac{(-1)^l}{(q-1)^l}$$

from which the statement follows. □

The result in this lemma can also be found in [56]. However, the proof in [56] uses a different approach based on a known result on the number of closed walks of length l in a complete graph of size q from a fixed but arbitrary vertex back to itself.

B.2 Proof of Theorem 6

We first prove that for all $(\mathsf{r_i}, \mathsf{r_o})$ pairs in $\mathscr{P}$ we have a positive normalized typical minimum distance. Then, we prove that this is not possible for any other $(\mathsf{r_i}, \mathsf{r_o})$ pair.

B.2.1 Proof of Sufficiency

A sufficient condition for a positive normalized typical minimum distance is

$$\lim_{\varpi \to 0^+} G(\varpi) < 0.$$

From Theorem 5 this is equivalent to

$$\mathsf{r_i}(1 - \mathsf{r_o}) > \lim_{\varpi \to 0^+} \max_{\lambda \in \mathscr{D}_\lambda} \mathsf{f}(\varpi, \lambda).$$

As we did in Lemma 1 and Lemma 2, let us use the notation $\varrho(\lambda) = \varrho_\lambda$ to emphasize the dependence on λ. We shall now show that

$$\lim_{\varpi \to 0^+} \max_{\lambda \in \mathscr{D}_\lambda} \mathsf{f}(\varpi, \lambda) = \max_{\lambda \in \mathscr{D}_\lambda} \lim_{\varpi \to 0^+} \mathsf{f}(\varpi, \lambda) = \max_{\lambda \in \mathscr{D}_\lambda} \left[\mathsf{r_i} \mathsf{H_b}(\lambda) + \log_2 (1 - \varrho(\lambda))\right]$$

that is, the maximization with respect to λ and the limit as $\varpi \to 0^+$ can be inverted, so that the region $\mathscr{P}$ in (6.13) is obtained.

This fact is proved by simply showing that

$$\lim_{\varpi \to 0^+} \mathsf{f}_{\max}(\varpi) = \mathsf{f}_{\max}(0),$$

that is, the function $\mathsf{f}_{\max}(\varpi) = \max_{\lambda \in \mathscr{D}_\lambda} \mathsf{f}(\varpi, \lambda)$ is right-continuous at $\varpi = 0$. For this purpose it suffices to show

$$\mathsf{f}_{\max}(\varpi) = \max_{\lambda \in (a,b)} \mathsf{f}(\varpi, \lambda) \tag{B.1}$$

where (a, b) is an interval independent of $\varpi \in [0, \frac{1}{2})$ such that the function

$$\log_2 \varrho(\lambda) - \log_2(1 - \varrho(\lambda))$$

is bounded over it, i.e.,

$$\sup_{\lambda\in(a,b)} |\log_2 \varrho_\lambda - \log_2(1-\varrho_\lambda)| = K\,.$$

Under these conditions we have uniform convergence of $\mathsf{f}(\varpi,\lambda)$ to $\mathsf{f}(0,\lambda)$ in the interval (a,b) as $\varpi \to 0^+$, namely,

$$\mathsf{f}(0,\lambda) - K\varpi \leq \mathsf{f}(\varpi,\lambda) \leq \mathsf{f}(0,\lambda) + K\varpi, \qquad \forall\lambda \text{ s.t. } a<\lambda<b\,. \tag{B.2}$$

The second inequality in (B.2) implies $\mathsf{f}_{\max}(\varpi) \leq \mathsf{f}_{\max}(0) + K\varpi$. Furthermore, denoting by $\hat{\lambda}\in(a,b)$ the maximizing λ, we have

$$\mathsf{f}_{\max}(0) - K\varpi = \mathsf{f}(0,\hat{\lambda}) - K\varpi \leq \mathsf{f}(\varpi,\hat{\lambda})$$

which implies $\mathsf{f}_{\max}(0) - K\varpi \leq \mathsf{f}_{\max}(\varpi)$. Hence, we have

$$\mathsf{f}_{\max}(0) - K\varpi \leq \mathsf{f}_{\max}(\varpi) \leq \mathsf{f}_{\max}(0) + K\varpi$$

that yields $\lim_{\varpi\to 0^+} \mathsf{f}_{\max}(\varpi) = \mathsf{f}_{\max}(0)$, as desired.

Next, we shall prove (B.1). We start by observing that in the case $\Omega_j = 0$ for all even j (in this case $\varrho(\lambda)$ is strictly increasing) by direct computation we have $\partial\,\mathsf{f}(\varpi,\lambda)/\partial\lambda < 0$ for all $0 \leq \varpi < 1/2$ and for all $1/2 \leq \lambda < 1$. Thus, in this case we can take $b = 1/2$. In all of the other cases there exists ξ such that $\varrho(\lambda) \leq \xi < 1$ for all $0<\lambda<1$ and we can take $b=1$. We prove the existence of $0<a<1/2$ (independent of $0\leq\varpi<1/2$) such that the maximum is not taken for all $0<\lambda\leq a$ as follows. Denoting $c = \log_2 e$ and $\varrho'(\lambda) = \mathrm{d}\varrho(\lambda)/\mathrm{d}\lambda$, we have

$$\frac{\partial\,\mathsf{f}(\varpi,\lambda)}{\partial\lambda} = \mathsf{r_i}\log_2(1-\lambda) - \mathsf{r_i}\log_2\lambda + c\,\varpi\,\frac{\varrho'(\lambda)}{\varrho(\lambda)} - c\,(1-\varpi)\frac{\varrho'(\lambda)}{1-\varrho(\lambda)}\,.$$

Given that $0<\varrho'(\lambda)<+\infty$ for all $0<\lambda\leq 1/2$ and since

$$\lim_{\lambda\to 0^+} \mathsf{r_i}(1-\varrho(\lambda))(\log_2(1-\lambda) - \log_2\lambda) = +\infty\,,$$

there exists $a > 0$ such that

$$\mathsf{r_i}(1 - \varrho(\lambda))(\log_2(1-\lambda) - \log_2 \lambda) > c\,\varrho'(\lambda), \qquad \text{for all } 0 < \lambda < a\,.$$

This latter inequality implies

$$\mathsf{r_i}(1 - \varrho(\lambda))(\log_2(1-\lambda) - \log_2 \lambda) > c\,\varrho'(\lambda) - \varpi \frac{c\,\varrho'(\lambda)}{\varrho(\lambda)}, \qquad \text{for all } 0 < \lambda < a$$

uniformly with respect to $\varpi \in [0, 1/2)$, that is equivalent to $\partial\,\mathsf{f}(\varpi, \lambda)/\partial\lambda > 0$ for all $0 < \lambda < a$, independently of $\varpi \in [0, 1/2)$. Hence, the maximum cannot be taken between 0 and a, with a independent of $\varpi \in [0, 1/2)$.

B.2.2 Proof of Necessity

So far it has been proved that the condition on $(\mathsf{r_i}, \mathsf{r_o})$ expressed by Theorem 6 is sufficient to have a positive normalized typical minimum distance. Now we need to show that this condition is also necessary. Concretely, we need to prove that for the ensemble $\mathscr{C}_\infty(\mathscr{C}_\mathsf{o}, \Omega, \mathsf{r_i}, \mathsf{r_o})$ all rate pairs $(\mathsf{r_i}, \mathsf{r_o})$ such that $\lim_{\varpi \to 0^+} G(\varpi) = 0$ (i.e., rate pairs on the boundary $\mathscr{P}$), the derivative of the growth rate at 0 is positive, $\lim_{\varpi \to 0^+} G'(\varpi) > 0$.

According to Lemma 1 the expression of the derivative of the growth rate, $G'(\varpi)$ corresponds to

$$G'(\varpi) = \log_2 \frac{1-\varpi}{\varpi} + \log_2 \frac{\varrho(\lambda_0)}{1 - \varrho(\lambda_0)}\,.$$

Therefore, since $G'(\varpi)$ is the sum of two terms the first of which diverges to $+\infty$ as $\varpi \to 0^+$, a necessary condition for the derivative to be negative is that the second term diverges to $-\infty$, i.e., $\lim_{\varpi \to 0^+} \varrho(\lambda_0) = 0$. This case is analyzed in the following lemma.

Lemma 4. *If $\varrho(\lambda) = 0$ then $\lambda \in \{0, 1\}$ in case the LT distribution Ω is such that $\Omega_j = 0$ for all odd j, and $\lambda = 0$ for any other LT distribution Ω.*

Proof. Let us recall that $\varrho(\lambda)$ is the probability that the LT encoder picks an odd number of nonzero intermediate bits (with replacement) given that the intermediate codeword has Hamming weight λh. If $\Omega_j > 0$ for at least one odd j, then the only case in which a zero LT encoded bit is generated with probability 1 is the one in which the intermediate word is the all-zero sequence. If $\Omega_j = 0$ for all odd j, there is also another case in which a nonzero bit is output by the LT encoder with probability 1, i.e., the case in which the intermediate word is the all-one word. □

Let us consider now a pair $(\mathsf{r_i}, \mathsf{r_o})$ such that $\lim_{\varpi \to 0^+} G(\varpi) = 0$. A fixed-rate Raptor code ensemble corresponding to this pair, has a positive typical minimum distance if and only if $\lim_{\varpi \to 0^+} G'(\varpi) < 0$. By Lemma 4 this implies $\lim_{\varpi \to 0^+} \lambda_0(\varpi) = 0$ when $\Omega_j > 0$ for at least one odd j. It implies either $\lim_{\varpi \to 0^+} \lambda_0(\varpi) = 0$ or $\lim_{\varpi \to 0^+} \lambda_0(\varpi) = 1$ otherwise. That $\lambda_0(\varpi)$ cannot converge to 0 follows from the proof of sufficiency (as shown, the maximum for $\varpi \in [0, 1/2)$ is taken for $\lambda > a > 0$). In order to complete the proof we now show that, in the case where $\Omega_j = 0$ for all odd j, assuming $\lim_{\varpi \to 0^+} \lambda_0(\varpi) = 1$ leads to a contradiction.

If $\Omega_j = 0$ for all odd j, a Taylor series for $\varrho(\lambda)$ around $\lambda = 1$ is $\varrho(\lambda) = \bar{\Omega}(1-\lambda) + o(\lambda)$. Assuming $\lim_{\varpi \to 0^+} \lambda_0(\varpi) = 1$, we consider the left-hand side of (6.12) and calculate its limit as $\varpi \to 0^+$. We obtain

$$\begin{aligned}
\lim_{\varpi \to 0^+} \frac{\partial \mathsf{f}}{\partial \lambda}(\varpi, \lambda_0) &= \mathsf{r_i} \lim_{\lambda_0 \to 1^-} \log_2 \frac{1-\lambda_0}{\lambda_0} + \lim_{\varpi \to 0^+} \left(\frac{\varpi}{\log 2} \frac{\varrho'(\lambda_0)}{\varrho(\lambda_0)} - \frac{1-\varpi}{\log 2} \frac{\varrho'(\lambda_0)}{1-\varrho(\lambda_0)} \right) \\
&= \mathsf{r_i} \lim_{\lambda_0 \to 1^-} \log_2 \frac{1-\lambda_0}{\lambda_0} + \frac{1}{\log 2} \lim_{\varpi \to 0^+} \frac{\varrho'(\lambda_0)(\varpi - \varrho(\lambda_0))}{\varrho(\lambda_0)(1-\varrho(\lambda_0))} \\
&= \mathsf{r_i} \lim_{\lambda_0 \to 1^-} \log_2 \frac{1-\lambda_0}{\lambda_0} + \frac{1}{\log 2} \lim_{\varpi \to 0^+} \frac{\bar{\Omega}(1-\lambda_0) - \varpi}{1-\lambda_0}
\end{aligned}$$

where the last equality follows from the above-stated Taylor series expansion. According to (6.12), the last expression must be equal to zero. This constraint requires the second limit to diverge to $+\infty$ (as the first limit diverges to $-\infty$). However, this cannot be fulfilled in any case when ϖ converges to zero and λ_0 to one. Actually, using standard Landau notation, when $1 - \lambda_0 = \Theta(\varpi)$ or $\varpi = \mathrm{o}(1-\lambda_0)$ the second limit converges, while when $1 - \lambda_0 = \mathrm{o}(\varpi)$ it diverges to $-\infty$.

B.3 Proof of Theorem 9

In this proof we derive first a lower bound for $G(\varpi)$ and then evaluate it for $\varpi \to 0^+$. To obtain a lower bound for $G(\varpi)$ we first derive a lower bound for $\mathcal{A}_\varpi$. Observing

(6.2) it can be seen how $\mathcal{A}_{\varpi}$ is obtained as a summation over all possible intermediate Hamming weights. A lower bound to $\mathcal{A}_{\varpi}$ can be obtained by limiting the summation to the term $\lambda^{\star} = 1 - \mathsf{r}_{\mathsf{o}}$ yielding to

$$\mathcal{A}_{\varpi n} \geq \frac{A^{\mathsf{o}}_{\lambda^{\star} h} A^{\mathsf{i}}_{\lambda^{\star} h, \varpi n}}{\binom{h}{\lambda^{\star} h}} = A^{\mathsf{o}}_{\lambda^{\star} h} \mathsf{Q}_{\varpi n, \lambda^{\star} h}$$

where we have introduced

$$\mathsf{Q}_{\varpi n, \lambda h} := \frac{A^{\mathsf{i}}_{\lambda h, \varpi n}}{\binom{h}{\lambda h}}$$

that represents the probability that the inner encoder outputs a codeword with Hamming weight ϖn given that the encoder input has weight λh.

Hence, we can now write

$$\begin{aligned} G(\varpi) &\geq \lim_{n \to \infty} \frac{1}{n} \log_2 A^{\mathsf{o}}_{\lambda^{\star} h} \mathsf{Q}_{\varpi n, \lambda^{\star} h} = \lim_{n \to \infty} \frac{1}{n} \log_2 A^{\mathsf{o}}_{\lambda^{\star} h} + \lim_{n \to \infty} \frac{1}{n} \log_2 \mathsf{Q}_{\varpi n, \lambda^{\star} h} \\ &= \mathsf{r}_{\mathsf{i}} \left(\mathsf{H}_{\mathsf{b}}(\lambda^{\star}) - (1 - \mathsf{r}_{\mathsf{o}}) \right) + \lim_{n \to \infty} \frac{1}{n} \log_2 \mathsf{Q}_{\varpi n, \lambda^{\star} h} \end{aligned} \tag{B.3}$$

We will now lower bound $\lim_{\varpi \to 0^+} \mathsf{Q}_{\varpi n, \lambda h}$. We denote by

$$q_{j,l} := \Pr\{X_i = 0 | w_{\mathsf{H}}(\mathbf{V}) = l, \deg(X_i) = j\}.$$

Note that $q_{j,l} = 1 - p_{j,l}$. We have that

$$\lim_{\varpi \to 0^+} \mathsf{Q}_{\varpi n, \lambda h} = \left(\sum_j \Omega_j q_{j, \lambda h} \right)^n \geq \left(\sum_j \Omega_j \underline{q}_{j, \lambda h} \right)^n$$

where $\underline{q}_{j,l} \leq q_{j,l}$ is the probability that the j intermediate symbols selected to encoder X_i are all zero. For large h, we have

$$\underline{q}_{j,l} = \left(1 - \frac{l}{h} \right)^j.$$

Denoting by $\underline{q}_l = \sum_j \Omega_j \underline{q}_{j,l}$, by Jensen's inequality we have

$$\underline{q}_l \geq \left(1 - \frac{l}{h}\right)^{\bar{\Omega}}.$$

Thus, we have that

$$\lim_{\varpi \to 0^+} \mathsf{Q}_{\varpi n, \lambda h} \geq (1-\lambda)^{n\bar{\Omega}}. \tag{B.4}$$

Replacing (B.4) in (B.3) and recalling that $h = n\mathsf{r_i}$ we obtain

$$\begin{aligned} G(\varpi) &\geq \mathsf{r_i}\left(\mathsf{H_b}(\lambda^\star) - (1-\mathsf{r_o})\right) + \lim_{n\to\infty} \frac{1}{n} \log_2 (1-\lambda^\star)^{n\bar{\Omega}} \\ &= \mathsf{r_i}\left(\mathsf{H_b}(\lambda^\star) - (1-\mathsf{r_o})\right) + \bar{\Omega} \log_2 (1-\lambda^\star) \\ &= \mathsf{r_i}\left(\mathsf{H_b}(1-\mathsf{r_o}) - (1-\mathsf{r_o})\right) + \bar{\Omega} \log_2 \mathsf{r_o}. \end{aligned}$$

By imposing $G(\varpi) = 0$ we obtain:

$$\phi(\mathsf{r_o}) = \frac{\bar{\Omega} \log_2(1/\mathsf{r_o})}{\mathsf{H_b}(1-\mathsf{r_o}) - (1-\mathsf{r_o})}.$$

This expression is only valid when the denominator is negative, that is, for $1 > \mathsf{r_o} > \mathsf{r_o^*}$, being $\mathsf{r_o^*}$ the only root of the denominator in $\mathsf{r_o} \in (0,1)$, whose approximate numerical value is $\mathsf{r_o^*} \approx 0.22709$.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[2] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Tech. J.*, vol. 29, no. 2, pp. 147–160, 1950.

[3] M. J. Golay, "Notes on digital coding," 1949.

[4] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Electronic Computers, Transactions of the IRE Professional Group on*, no. 3, pp. 6–12, 1954.

[5] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Information Theory, Transactions of the IRE Professional Group on*, vol. 4, no. 4, pp. 38–49, 1954.

[6] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (paris)*, vol. 2, no. 147-156, p. 116, 1959.

[7] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[8] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[9] D. J. Costello and G. D. Forney, "Channel coding: The road to channel capacity," *Proc. IEEE*, vol. 95, pp. 1150–1177, June 2007.

[10] P. Elias, "Coding for two noisy channels," in *In Proc. of Inf. Theory: Third London Symp.*, pp. 61–74, London: Butterworth Scientific, Ed. C. Cherry, 1955.

[11] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Trans. Inform. Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[12] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[13] G. D. Forney Jr, *Concatenated codes.* Cambridge, MA: MIT Press, 1966.

[14] Consulative Committee for Space Data Systems (CCSDS), “Telemetry channel coding,” *Silver Book, CCSDS 101.0-B-1-S*, May 1984.

[15] C. Berrou and A. Glavieux, “Near optimum error correcting coding and decoding: Turbo-codes,” *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[16] R. G. Gallager, *Low-Density Parity-Check Codes.* PhD thesis, Dep. Electrical Eng., M.I.T, Cambridge, MA, July 1963.

[17] D. J. C. MacKay and R. M. Neal, “Near shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 33, pp. 457–458, Mar. 1997.

[18] J.Metzner, “An improved broadcast retransmission protocol,” *IEEE Trans. Commun.*, vol. 32, pp. 679–683, June 1984.

[19] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical loss-resilient codes,” in *Proc. 29th Symp. Theory Computing*, pp. 150–159, 1997.

[20] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[21] P. Oswald and A. Shokrollahi, “Capacity-achieving sequences for the erasure channel,” *IEEE Trans. Inform. Theory*, vol. 48, pp. 3017–3028, Dec. 2002.

[22] D. Burshtein and G. Miller, “An efficient maximum likelihood decoding of LDPC codes over the binary erasure channel,” *IEEE Trans. Inform. Theory*, vol. 50, nov 2004.

[23] E. Paolini, G. Liva, B. Matuz, and M. Chiani, “Maximum likelihood erasure decoding of LDPC codes: Pivoting algorithms and code design,” *IEEE Trans. Commun.*, vol. 60, pp. 3209–3220, Nov. 2012.

[24] J. Byers, M. Luby, M. Mitzenmacher, and A.Rege, “A digital fountain approach to reliable distribution of bulk data,” in *Proc. of ACM SIGCOMM*, 1998.

[25] M. Luby, “LT codes,” in *Proc. 43rd Annual IEEE Symp. on Foundations of Computer Science*, (Vancouver, Canada), pp. 271–282, Nov. 2002.

[26] A. Shokrollahi and M. Lassen, S. Luby, “Multi-stage code generator and decoder for communication systems,” Dec. 2001. US Patent 7,068,729.

[27] A. Shokrollahi, “Raptor codes,” in *Proc. of the 2004 IEEE Int. Symp. on Inf. Theory*, (Chicago, Illinois, US), p. 36, June 2004.

[28] M. Shokrollahi, “Raptor codes,” *IEEE Trans. Inform. Theory*, vol. 52, pp. 2551–2567, June 2006.

[29] P. Maymounkov, “Online codes,” tech. rep., Technical report, New York University, 2002.

[30] 3GPP TS 26.346 V11.1.0, “Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs,” June 2012.

[31] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, “RFC 5053: Raptor forward error correction scheme: Scheme for object delivery,” tech. rep., IETF, Oct. 2007.

[32] M. Shokrollahi, S. Lassen, and R. Karp, “Systems and processes for decoding chain reaction codes through inactivation,” Feb. 2005. US Patent 6,856,263.

[33] F. Lázaro Blasco, G. Liva, and G. Bauch, “LT code design for inactivation decoding,” in *Proc. 2014 IEEE Inf. Theory Workshop*, (Hobart, Tasmania, Australia), pp. 441–445.

[34] F. Lázaro Blasco, G. Liva, and G. Bauch, “Enhancing the LT component of Raptor codes,” in *Proc. of the 10th Int. ITG Conf. on Systems, Commun. and Coding, SCC 2015*, (Hamburg, Germany).

[35] F. Lázaro, G. Liva, and G. Bauch, “Inactivation decoding analysis for LT codes,” in *Proc. 52nd Annu. Allerton Conf. on Commun., Control, and Computing*, (Monticello, Illinois, USA), Oct. 2015.

[36] F. Lázaro Blasco and G. Liva, “On the concatenation of non-binary random linear fountain codes with maximum distance separable codes,” in *Proc. 2011 IEEE Int. Conf. on Commun., (ICC)*, (Kyoto, Japan).

[37] F. Lázaro Blasco, , G. Garrammone, and G. Liva, “Parallel concatenation of non-binary linear random fountain codes with maximum distance separable codes,” *IEEE Trans. Commun.*, vol. 61, pp. 4067–4075, Oct. 2013.

[38] F. Lázaro Blasco, E. Paolini, G. Liva, and G. Bauch, “On the weight distribution of fixed-rate Raptor codes,” in *Proc. of the 2015 IEEE Int. Symp. on Inf. Theory*, (Hong Kong, China), pp. 2880–2884, June 2015.

[39] F. Lázaro, E. Paolini, G. Liva, and G. Bauch, “Distance spectrum of fixed-rate Raptor codes with linear random precoders,” *IEEE J. Select. Areas Commun.*, Dec. 2015.

[40] F. Lázaro, G. Liva, E. Paolini, and G. Bauch, “Bounds on the error probability of Raptor codes,” (Washington DC, USA), Dec. 2016. Proc. IEEE Globecom, to be published, availabe online under *http://arxiv.org/abs/1604.07560.*

[41] G. Garrammone and F. Lázaro Blasco, “On fragmentation for fountain codes,” in *Proc. of the 10th Int. ITG Conf. on Systems, Commun. and Coding, SCC 2013*, (Munich, Germany).

[42] J. L. Massey, “Capacity, cutoff rate, and coding for a direct-detection optical channel,” *IEEE Trans. Commun.*, vol. 29, pp. 1615–1621, Nov. 1981.

[43] R. Singleton, “Maximum distance q-nary codes,” *IEEE Trans. Inform. Theory*, vol. 10, no. 2, pp. 116–118, 1964.

[44] E. Berlekamp, “The technology of error-correcting codes,” *IEEE Proceedings*, vol. 68, pp. 564–593, 1980.

[45] E. Berlekamp, R. McEliece, and H. Vantilborg, “On the inherent intractability of certain coding problems,” *IEEE Trans. Inform. Theory*, vol. 24, pp. 384–386, 1978.

[46] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570 –1579, jun 2002.

[47] G. Liva, E. Paolini, and M. Chiani, "Bounds on the error probability of block codes over the q-ary erasure channel," *IEEE Trans. Commun.*, vol. 61, pp. 2156–2165, June 2013.

[48] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed., 1994.

[49] T. M. Cover and J. A. Thomas, *Elements of information theory.* New York: Wiley, 2nd ed., 2006. chapter 15.

[50] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for Network Coding," in *ISIT'08: Proc. of the 2009 IEEE Int. Symp. on Inf. Theory*, pp. 1651–1655, July 2008.

[51] G. Liva, E. Paolini, and M. Chiani, "Performance versus overhead for fountain codes over $\mathbb{F}_q$," *IEEE Comm. Letters.*, vol. 14, no. 2, pp. 178–180, 2010.

[52] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. 2004 IEEE Int. Symp. on Inf. Theory*, (Chicago, Illinois, US), June 2004.

[53] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," in *Proc. 2006 IEEE Int. Symp. on Inf. Theory*, (Seattle, Washington, US), pp. 2677–2679, 2006.

[54] G. Maatouk and A. Shokrollahi, "Analysis of the second moment of the LT decoder," *IEEE Trans. Inform. Theory*, vol. 58, pp. 2558–2569, May 2012.

[55] A. Shokrollahi, "Theory and applications of Raptor codes," *Mathknow*, vol. 3, pp. 59–89, 2009.

[56] B. Schotsch, G. Garrammone, and P. Vary, "Analysis of LT codes over finite fields under optimal erasure decoding," *IEEE Commun. Lett.*, vol. 17, pp. 1826–1829, Sept. 2013.

[57] D. H. Wiedemann, "Solving sparse linear equations over finite fields," *IEEE Trans. Inform. Theory*, vol. 32, pp. 54–62, Jan. 1986.

[58] B. A. LaMacchia and A. M. Odlyzko, “Solving large sparse linear systems over finite fields,” *Advances in Cryptology-CRYPT0’90*, pp. 109–133, 1991.

[59] M. Shokrollahi and M. Luby, “Systematic encoding and decoding of chain reaction codes,” June 2005. US Patent 6,909,383.

[60] A. Shokrollahi and M. Luby, “Raptor codes,” *Foundations and Trends in Commun. and Inf. Theory*, vol. 6, no. 3-4, pp. 213–322, 2011.

[61] RFC 5053, “Network working group; Request for Comments: 5053; Raptor Forward Error Correction Scheme for Object Delivery,” Oct. 2007.

[62] ETSI TR 102 993 V1.1.1, “Digital Video Broadcasting (DVB); Upper Layer FEC for DVB Systems,” Feb. 2011.

[63] ETSI TS 102 472 V1.3.1, “Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols,” June 2009.

[64] ETSI TS 102 034 V1.5.1, “Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks,” May 2014.

[65] ETSI EN 301 790 V1.5.1, “Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems,” May 209.

[66] ITU-T H.701, “International Telecommunication Union (ITU); Series H: Audiovisual and Multimedia Systems; IPTV multimedia services and applications for IPTV - General aspects; Content delivery error recovery for IPTV services,” Mar. 2009.

[67] T.-C. Ng and S. Yang, “Finite-length analysis of BATS codes,” in *Proc. of 2013 IEEE Int. Symp. on Network Coding, (NetCod)*, (Calgary, Alberta, Canada), June 2013.

[68] K. Mahdaviani, M. Ardakani, and C. Tellambura, “On Raptor code design for inactivation decoding,” *IEEE Commun. Lett.*, vol. 60, pp. 2377–2381, Sept. 2012.

[69] S. Kirkpatrick, D. Gelatt, and M. Vecchi, “Optimization by simmulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[70] N. Rahnavard, B. Vellambi, and F. Fekri, “Rateless codes with unequal error protection property,” *IEEE Trans. Inform. Theory*, vol. 53, pp. 1521–1532, Apr. 2007.

[71] B. E. Schotsch, *Rateless Coding in the Finite Length Regime.* PhD thesis, Inst. of Commun. Systems and Data Proc., RWTH Aachen, Aachen, Germany, July 2014.

[72] P. Wang, G. Mao, Z. Lin, M. Ding, W. Liang, X. Ge, and Z. Lin, “Performance analysis of Raptor codes under maximum likelihood decoding,” *IEEE Trans. Commun.*, vol. 64, pp. 906–917, Mar. 2016.

[73] F. Mac Williams and N. Sloane, *The theory of error-correcting codes*, vol. 16. North Holland Mathematical Libray, 1977.

[74] A. Ashikhmin and A. Barg, “Minimal vectors in linear codes,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 2010–2017, Sept. 1998.

[75] A. Barg, J. Justesen, and C. Thommesen, “Concatenated codes with fixed inner code and random outer code,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 361–365, Jan. 2001.

[76] A. Orlitsky, K. Viswanathan, and J. Zhang, “Stopping set distribution of LDPC code ensembles,” *IEEE Trans. Inform. Theory*, vol. 51, pp. 929–953, Mar. 2005.

[77] C. Di, T. Richardson, and R. Urbanke, “Weight distribution of low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 52, pp. 4839–4855, Nov. 2006.

[78] K. Kasai, D. Declercq, and K. Sakaniwa, “Fountain coding via multiplicatively repeated non-binary LDPC codes,” *IEEE Trans. Commun.*, vol. 60, pp. 2077–2083, Aug. 2012.

[79] F. Chiaraluce and R. Garello, “On the asymptotic performance of Hamming product codes,” in *Proc. 6th Int. Symp. on Commun. Theory and Applications*, pp. 329—334, 2001.

[80] A. Barg and G. D. Forney, “Random codes: minimum distances and error exponents,” *IEEE Trans. Inform. Theory*, vol. 48, pp. 2568–2573, Sept. 2002.

Curriculum Vitae

Last name:	Lázaro Blasco
First name:	Francisco
Nationality:	Spanish
Date of birth:	24.03.1983
Place of birth:	Zaragoza, Spain

09.1989 - 06.1997 Primary school in Alcañiz, Teruel, Spain

09.1997 - 06.1999 Secondary school in Alcañiz, Teruel, Spain

09.1999 - 06.2001 High school in Alcañiz, Teruel, Spain

09.2001 - 12.2006 Studies of *Telecommunication Engineering* at the University of Zaragoza, Spain
Degree: Ingeniero Superior de Telecomunicaciones

11.2006 - 03.2007 Intern at Siemens Networks in Munich, Germany

04.2007 - 04.2008 Test Engineer at Rohde & Schwarz in Munich, Germany

07.2008 - present Scientific researcher in satellite and space communications at German Aerospace Center (DLR) in Oberpfaffenhofen, Germany