ANR, Appel à Projets Générique (AAPG 2019)

QCSP Project (ANR-19-CE25-0013-02)

Deliverable D1.2a

Intermediate report on the design of low rate non-binary codes

Editor:	Charbel Abdel Nour (IMT Atlantique)
Deliverable nature:	Internal (scope: Consortium and ANR)
Due date:	October 3, 2020
Delivery date:	October 5, 2020
Version:	1.0
Total number of pages:	37 pages
Keywords:	Non-binary codes, code design, Turbo codes, LDPC codes, Polar codes

Abstract
This deliverable is intended to first recall the design constraints of non-binary codes stemming from D1.1 and agreed upon. Designed codes are selected from 3 different code families: Turbo codes, Low-Density Parity Check (LDPC) codes and Polar codes. Then for each code family, the design procedure for the non-binary code to be associated with the CCSK modulation will be described. Initial results based on the design procedure and associated with a BPSK/QPSK constellation are reported next. These results may include effects of some of the design parameters that are specific to each code family.

List of Authors

Partner	Author
LAB-STICC/UBS	Emmanuel Boutillon (Emmanuel.Boutillon@univ-ubs.fr), Cédric Marchand (Cedric.marchand@univ-ubs.fr)
CEA-LETI	Valentin Savin (valetin.savin@cea.fr)
IMT Atlantique	Charbel Abdel Nour (charbel.abdelnour@imt-atlantique.fr), Rami Klaimi (Rami.Klaimi@imt-atlantique.fr)

Contents

Executive Summary

Work Package 1 (WP1) of the QCSP project is concerned with the design and selection of best candidates of non-binary codes to be associated with the CCSK modulation. It is divided into 4 tasks reported in 4 deliverables as can be seen in figure 1:

- Task 1.1 intended to define all required parameters for the design.
- Task 1.2 where the actual code design takes place.
- Task 1.3 dedicated for complexity assessment and reduction for the different proposals.
- Task 1.4 that aims to compare and validate final solutions.

The current D1.2a deliverable is provided in the context of **Task 1.2**. It is intended to specify design constraints stemming from T1.1 and D1.1, then to provide the design methods for each considered code family followed by obtained initial results regarding performance and the impact of some design parameters. An updated version D1.2b of this deliverable is expected at a later stage with the final results.

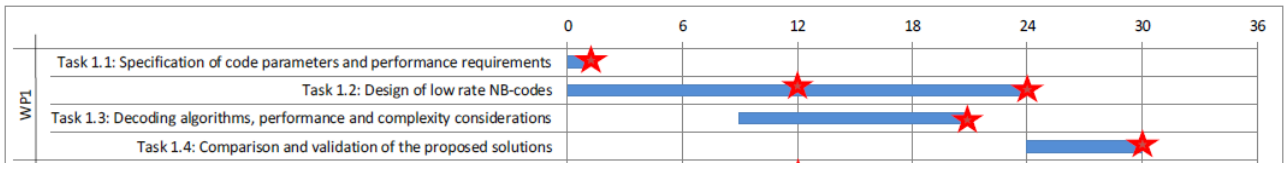


Figure 1: Gantt diagram of WP1

This deliverable is organized as follows:

Section 1 recalls the design constraints/parameters provided by D1.1.

Section 2 is dedicated to non-binary Turbo codes. The first sub-section describes the design method. The second subsection provides simulation results including the effect of specific parameters.

Section 3 is dedicated to non-binary LDPC codes. The first sub-section describes the design method. The second subsection provides simulation results including the effect of specific parameters.

Section 4 is dedicated to non-binary Polar codes. The first sub-section describes the design method. The second subsection provides simulation results including the effect of specific parameters.

Section 5 concludes the deliverable.

1 Design constraints and parameters

1.1 Code parameters and performance requirements

Task 1.1 was devised to define different system constraints and parameters. These were reported in D1.1. Having the code design as a target, this section starts by recalling relevant parameters stemming from D1.1. The following Non-Binary (NB) code parameters have been agreed on by the QCSP partners during the project kickoff meeting on October 3rd, 2019.

- Information block size (K)
  - $K = 120, 240, 480, 960, 1920$ bits.
  - For NB codes defined over an alphabet of size $q = 2^p$, the above values must be approximated to the nearest multiple of p, if needed (they have been selected such that they are multiple of $p = 6, 8, 10, 12$, except for $K = 120$ which is only a multiple of $p = 6, 10, 12$).
- Native coding rate (R) of the NB code
  - $R = 1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3$.
  - Incremental redundancy is optional, but recommended.
- Coded block size (N)
  - Any combination of the above K and R values may be considered, provided that the coded block size $N = K/R$ does not exceed the maximum value N_{max}= 24000 bits. This means that lowest coding rates start to be disregarded from $K = 960$ bits frame size.
- Channel model
  - Additive White Gaussian Noise (AWGN) channel, with bipolar ± 1 input.
  - 3GPP channel model, to be determined later.
- Target Frame Error Rate (FER) level
  - Over AWGN channel, a FER of 10^{-3} to 10^{-4} is targeted.

The above parameters may not be suitable for some of the three NB code families. For instance, for a NB Polar code, the mother NB code length is a power of two. Thus, shortening or puncturing should be used just for the sake of adapting to the proposed parameters, although this might not be the most practical solution. Each code designer should adapt his design to suit best the parameters provided above.

2 Design of non-binary turbo codes

2.1 Introduction

The advantages of the use of NB codes with CCSK modulation was clearly identified in D1.1. One of the candidate families of codes that can be used are turbo codes. Recently, several studies have targeted the design of NB turbo codes (NB-TC). In [1], a NB-TC structure was derived from a protograph sub-ensemble of a NB-LDPC code. In [2], NB-TCs over GF(q) were designed based on their EXtrinsic Information Transfer (EXIT) charts and an estimate of their Truncated Union Bounds (TUB). Both studies report significant gains in the short-to-moderate block size regime compared to state-of-the-art binary Forward Error Correcting (FEC) codes.

However our latest work performed on the subject has shown even a larger potential for the NB-TC family. In fact, all the constituent components have been addressed one-by-one, largely improving the end result. The design procedure will be briefly recalled next.

2.2 Design method/procedure

2.2.1 Component codes

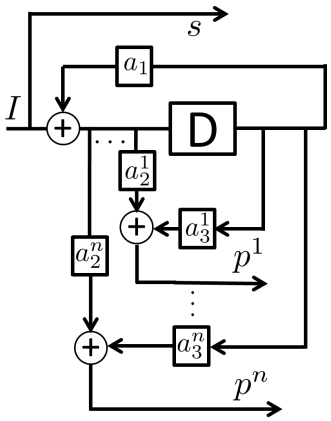


Figure 2.1: NB convolutional code structure with n different generated parities.

The designed NB-TCs consist of a parallel concatenation of two memory-1 NB Convolutional Codes (CC) and a symbol-wise interleaver followed by a puncturer for rate adaptation. The component NB-CCs are designed following the improved accumulator template of figure 2.1 originally proposed in [3]. To define a base code with rate $R = 1/3$, the triplet (a_1, a_2^1, a_3^1) needs to be defined. Lower rates can be achieved by specifying additional parity polynomials p^i. The adopted procedure is the one provided in [3]. This structure, coupled with the proposed design procedure were shown to improve the cumulated Euclidean distance by up to 25% compared to the classical accumulator structure [3].

While this increase in distance may be largely beneficial when targeting low error rates, it may introduce a performance penalty at high error rates. Having a target FER of around 10^{-4} in QCSP, a compromise needs to be found to achieve best convergence behaviour while satisfying the target error rate. It can be reached thanks to the multitude of parameters to tweak through the values of the triplet (a_1, a_2^1, a_3^1).

The output of the code defined over GF(q) is mapped to a 4-Quadrature Amplitude Modulation (QAM) constellation. In this case, q has to be a power of 4 and the q-ary symbols are decomposed into 4-ary symbols.

2.2.2 Interleaver design

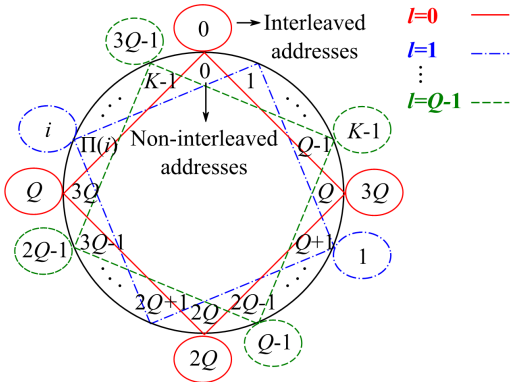


Figure 2.2: Possible representation of the ARP interleaver structure, with $S(0), \ldots, S(Q-1) = 0$ and $K = 4Q$, when using circular termination.

As introduced by Berrou et al. [4], the minimum Hamming distance (d_{min}) of a TC is not only defined by its constituent encoders, but also fixed through the TC interleaver. the proposal is based on several technical contributions regarding the joint design of interleavers and puncturing patterns addressing the improvement in performance at low error rates, on the one hand, and for short frame sizes, on the other hand [5,6]. Specifically, a new puncturing constraint related to parity puncturing is proposed for the design of TC interleavers. Trellis termination was also modified to avoid the spectral efficiency loss of tail symbols. The work focused on the Almost Regular Permutation (ARP) [7, 8] structure derived from the regular interleaver structure with the introduction of a degree of disorder through a vector of shifts S:

$$\pi(i) = (P.i + S(i mod Q)) mod K \tag{2.1}$$

where P is the interleaver period that must be relatively prime to K. *mod* represents the modulo operation. The shift vector S has length Q and to guarantee the bijectivity of the ARP interleaver function, Q must be a divisor of K. Two main criteria have to be considered for the design of a TC interleaver: the Hamming distance spectrum of the TC and the correlation between the channel information and extrinsic data at the input of each component decoder. Two measurable parameters related to these criteria have been considered: minimum span S_{min} [7] and correlation girth g [9]. It can be shown that:

$$\pi(i + Q) mod Q = \pi(i) mod Q \tag{2.2}$$

Therefore, Q groups of permutation addresses are identified in the whole interleaver size K, each one corresponding to a given modulo Q value. Thus, the interleaver parameters can be selected by dividing the permutation addresses into Q groups or layers with index $l = 0, ..., Q-1$. The final permutation addresses of each layer l are defined by choosing its corresponding shift value $S(l)$. The layer number l to which address i in the interleaved sequence d' belongs is determined as $l = i mod Q$. For each layer l, we propose to derive the shift value by:

$$S(l) = T_l + A_l.Q \tag{2.3}$$

where $T_l = 0, \ldots, Q-1$ and $A_l = 0, \ldots, K/Q-1$ correspond to the inter and intra-layer shifts for layer l, respectively. In figure 2.2, the interleaver structure is divided into its Q corresponding layers and represented in a circle with the addresses of d (non-interleaved) and d' (interleaved) in its inner and outer parts, respectively as shown in figure 2.2.

2.2.3 Puncturing mask selection

A periodic puncturing pattern with period M is considered. Different Circular Recursive Systematic Convolutional (CRSC) codes with native coding rate R dependent on the target rate are used for the

two constituent codes of the TC. Thus, the puncturing mask is composed of two vectors of length M, corresponding to the puncturing positions in the data and parity vectors. The puncturing mask is defined according to the target code rate R of the TC and to the puncturing period M.

The extrinsic information computed from non-punctured parity positions is expected to be more reliable than the one generated from punctured parity positions. Since extrinsic information is used as a priori information on data, a possible strategy for the interleaver construction involves connecting the positions with highly reliable extrinsic information to the positions with less reliable extrinsic information, which are more prone to errors. This connection strategy aims to spread the correction capability of the TC over the whole data block. The puncturing mask design procedure associated with the interleaver connection proposed in this work involves the following steps:

- Find the best puncturing pattern: The best puncturing mask is the one generating the best CRSC Hamming distance spectrum.
- Identify error-prone systematic positions in a puncturing period M: To this end, systematic puncturing is introduced and the resulting CRSC distance spectrum is evaluated. The punctured systematic positions leading to the poorest CRSC distance spectrum (i.e., the lower d_{min} and the higher number of codewords at that distance or multiplicity) are then labeled as the most error-prone. Note that systematic puncturing is only introduced to identify error-prone systematic positions and is then removed from the puncturing mask.
- Apply the connection strategy: The proposed strategy involves connecting non-punctured parity positions of a CRSC code to the most error-prone positions of the other one.

In the ARP model, puncturing constraints are included via the T_l values. To simplify their inclusion, Q is set as a multiple of M (in this study, $Q = M$). Since the layer order of the ARP interleaver is Q-periodic, the validation of puncturing constraints in a puncturing period M is a sufficient condition for their validation in the whole data sequence. In the following sections, an interleaver including the proposed parity puncturing constraint is used.

2.2.4 Fine-grained code rate flexibility and Hybrid Automatic Repeat Request (HARQ) support

Contrary to the classical method for designing rate-compatible TCs where the design starts from the mother code rate and progressively punctures symbols to obtain higher coding rates, we propose to design the interleaver with the underlying connection strategy specifically optimized for the highest coding rate to be supported. Indeed, this guarantees the best performance for this highest coding rate, avoiding high error floors. Then, positions are progressively removed in the puncturing mask, generating more parities and lowering the coding rate. These positions are selected following a comparative study of obtained distance spectra of all remaining candidate positions in the mask for the target code rate. By applying M-periodic puncturing patterns, the codeword granularity is naturally $1/M$ when one symbol is added in each of the parity puncturing masks. After defining the incremental hierarchy to remove the punctured positions in the M-periodic mask, $M-1$ coding rates are now supported incrementally with U additional parity symbols, corresponding to the number of M-periods in the information frame size K when moving from one code rate to the next ($U = [K/M]$ where $[x]$ represents the integer part of x). The code rate of the TC can be expressed as:

$$R = M/(M + 2N_{np}) \tag{2.4}$$

Where N_{np} represents the number of non-punctured parity symbols per constituent code in the mask of period M. If need be, we propose to support codeword granularities lower than $1/M$ by removing the designated punctured positions in a subset of the U periods. This subset is spread uniformly within the U possibilities. When diminishing the subset size, a codeword granularity down to one symbol can be supported. The performance of the resulting code is expected to naturally lie between the surrounding larger granularity (of $1/M$) code rates.

2.3 Simulation results

Fame size in bits	Frame size in GF(64) symbols	Frame size in GF(256) symbols
120	20	15
240	40	30
480	80	60
960	160	120
1920	320	240

Table 2.1: Frame sizes in bit and symbols.

Table 2.1 converts the target frame sizes agreed upon in D1.1 to their equivalent values in number of GF(q) symbols, where $q = 64$ and 256. These are used as identifiers for the design of the interleaver.

GF(64)	GF(256)
$x^6 + x^5 + x^4 + x + 1$	$x^8 + x^4 + x^3 + x^2 + 1$

Table 2.2: Primitive polynomials.

Table 2.2 provides the primitive polynomials used over GF(64) and GF(256) respectively.

GF(64)			
Frame size in symbols	P	Q	$(S(0), ..., S(Q-1))$
20	3	4	(0,7,9,16)
40	9	4	(0,22,36,34)
80	71	4	(0,70,29,61)
160	143	8	(0,150,62,10,110,22,20,98)
320	151	8	(0,311,67,30,240,71,67,254)
GF(256)			
Frame size in symbols	P	Q	$(S(0), ..., S(Q-1))$
15	11	3	(0,9,6)
30	19	3	(0,25,14)
60	17	4	(0,4,16,44)
120	109	8	(0,109,103,76,115,108,38,19)
240	127	8	(0,220,30,168,58,236,136,96)

Table 2.3: ARP interleaver parameters.

Table 2.3 provides the ARP protograph interleaver parameters following equation 2.1. In particular, the values of parameter P, the period of the interleaver Q, and the values of the shift vector S of length Q for the considered frame sizes and GF(64) and GF(256) for an association with a 4-QAM constellation.

Table 2.4 provides the generator polynomials or the parameters of the triplet (a_1, a_2^1, a_3^1) and the additional couple (a_x^i, a_y^i). C_1 denotes the best code, in the sense of the obtained accumulated minimum Hamming distance of the NB-CC, whereas C_2 denotes the worst code in the same sense. Performance boundaries or extreme edges should lie within the combination of the TC with C_1 as component and the one with C_2 as component code. However for the target error rate of QCSP, the TC formed by the combination of the C_1 and the C_2 code seem to provide a good balance between low and high error rate performance level.

For all performed simulations, we have used the scaled simplified Max-Log MAP decoding with 8 iterations. Simulations were performed over an AWGN channel with 4-QAM modulation. Figures 2.3

Coefs	GF(64)		GF(256)	
	C_1	C_2	C_1	C_2
(a_1, a_2^1, a_3^1)	$(15, 25, 36)$	$(24, 29, 0)$	$(15, 98, 108)$	$(15, 84, 0)$
(a_2^2, a_3^2)	$(48, 29)$	$(29, 7)$	$(177, 194)$	$(1, 1)$
(a_2^3, a_3^3)	$(47, 16)$	$(29, 58)$	$(42, 102)$	$(1, 2)$
(a_2^4, a_3^4)	$(30, 38)$	$(29, 29)$	$(105, 174)$	$(2, 1)$
(a_2^5, a_3^5)	$(5, 14)$	$(58, 29)$	$(45, 252)$	$(1, 4)$
(a_2^6, a_3^6)	$(7, 13)$	$(7, 29)$	$(58, 96)$	$(4, 1)$
(a_2^7, a_3^7)	$(22, 45)$	$(14, 29)$	$(170, 201)$	$(1, 8)$
(a_2^8, a_3^8)	$(32, 56)$	$(29, 14)$	$(37, 237)$	$(8, 1)$
(a_2^9, a_3^9)	$(21, 27)$	$(28, 29)$	$(26, 35)$	$(1, 16)$
(a_2^{10}, a_3^{10})	$(10, 49)$	$(29, 28)$	$(106, 162)$	$(16, 1)$
(a_2^{11}, a_3^{11})	$(35, 43)$	$(7, 26)$	$(161, 217)$	$(1, 32)$
(a_2^{12}, a_3^{12})	$(9, 50)$	$(7, 39)$	$(28, 151)$	$(32, 1)$
(a_2^{13}, a_3^{13})	$(3, 62)$	$(7, 61)$	$(235, 254)$	$(32, 142)$
(a_2^{14}, a_3^{14})	$(42, 49)$	$(26, 7)$	$(9, 20)$	$(142, 32)$
(a_2^{15}, a_3^{15})	$(12, 29)$	$(26, 29)$	$(78, 159)$	$(1, 3)$
(a_2^{16}, a_3^{16})	$(46, 61)$	$(26, 58)$	$(139, 208)$	$(1, 5)$
(a_2^{17}, a_3^{17})	$(37, 60)$	$(29, 26)$	$(27, 99)$	$(1, 6)$
(a_2^{18}, a_3^{18})	$(4, 54)$	$(29, 39)$	$(120, 248)$	$(1, 9)$
(a_2^{19}, a_3^{19})	$(24, 42)$	$(29, 61)$	$(32, 179)$	$(1, 10)$
(a_2^{20}, a_3^{20})	$(11, 63)$	$(39, 7)$	$(2, 155)$	$(1, 12)$
(a_2^{21}, a_3^{21})	$(5, 41)$	$(39, 29)$	$(55, 200)$	$(1, 17)$
(a_2^{22}, a_3^{22})	$(44, 52)$	$(39, 58)$	$(116, 169)$	$(1, 18)$
(a_2^{23}, a_3^{23})	$(17, 28)$	$(58, 26)$	$(61, 68)$	$(1, 20)$
(a_2^{24}, a_3^{24})	$(16, 40)$	$(61, 29)$	$(44, 205)$	$(1, 24)$

Table 2.4: Generator polynomials of the best (C_1) and worst (C_2) NB-CCs over GF(64) and GF(256).

and 2.4 show the performance results of the codes designed over GF(64) and GF(256) respectively for the information frame size K=120 bits and coding rates ranging from R=1/48 up to R=2/3. Figures 2.5 and 2.6 show the performance results of the codes designed over GF(64) and GF(256) respectively for the information frame size K=240 bits and coding rates ranging from R=1/48 up to R=2/3. Figures 2.7 and 2.8 show the performance results of the codes designed over GF(64) and GF(256) respectively for the information frame size K=480 bits and coding rates ranging from R=1/48 up to R=2/3. Figures 2.9 and 2.10 show the performance results of the codes designed over GF(64) and GF(256) respectively for the information frame size K=960 bits and coding rates ranging from R=1/24 up to R=2/3. Finally, figures 2.11 and 2.12 show the performance results of the codes designed over GF(64) and GF(256) respectively for the information frame size K=1920 bits and coding rates ranging from R=1/12 up to R=2/3.

All results show that the NB-TC made of the combination of C_1 and C_2 can improve performance of up to 0.5dB without showing any change in the slope of the curves down to 10^{-4} of FER. Therefore they provide a better tradeoff between performance at low and high error rates. We can also notice that the performance of GF(256) codes is slightly worse than the one of GF(64) codes. However, they offer a largely higher Hamming distance. These results may differ when associated with a CCSK modulation planned for the next version of the deliverable.

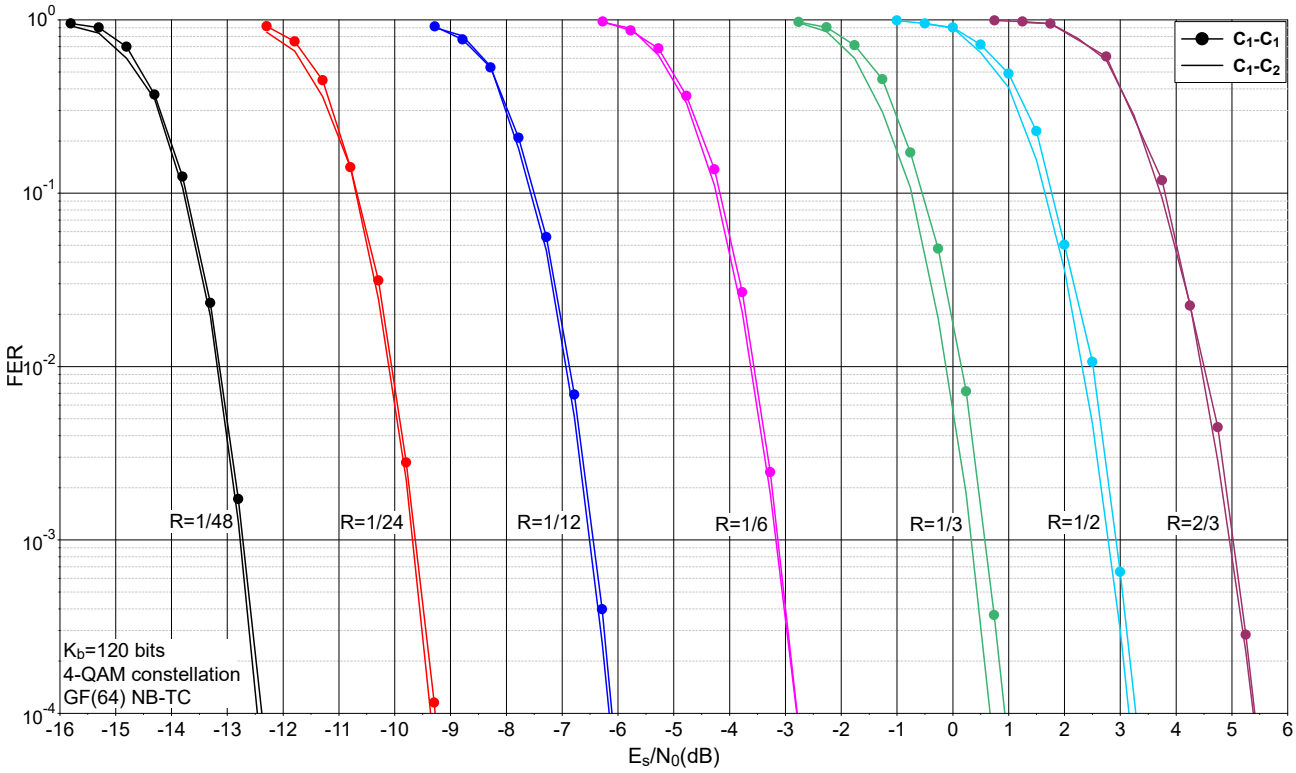


Figure 2.3: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(64), information frame size K=120 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

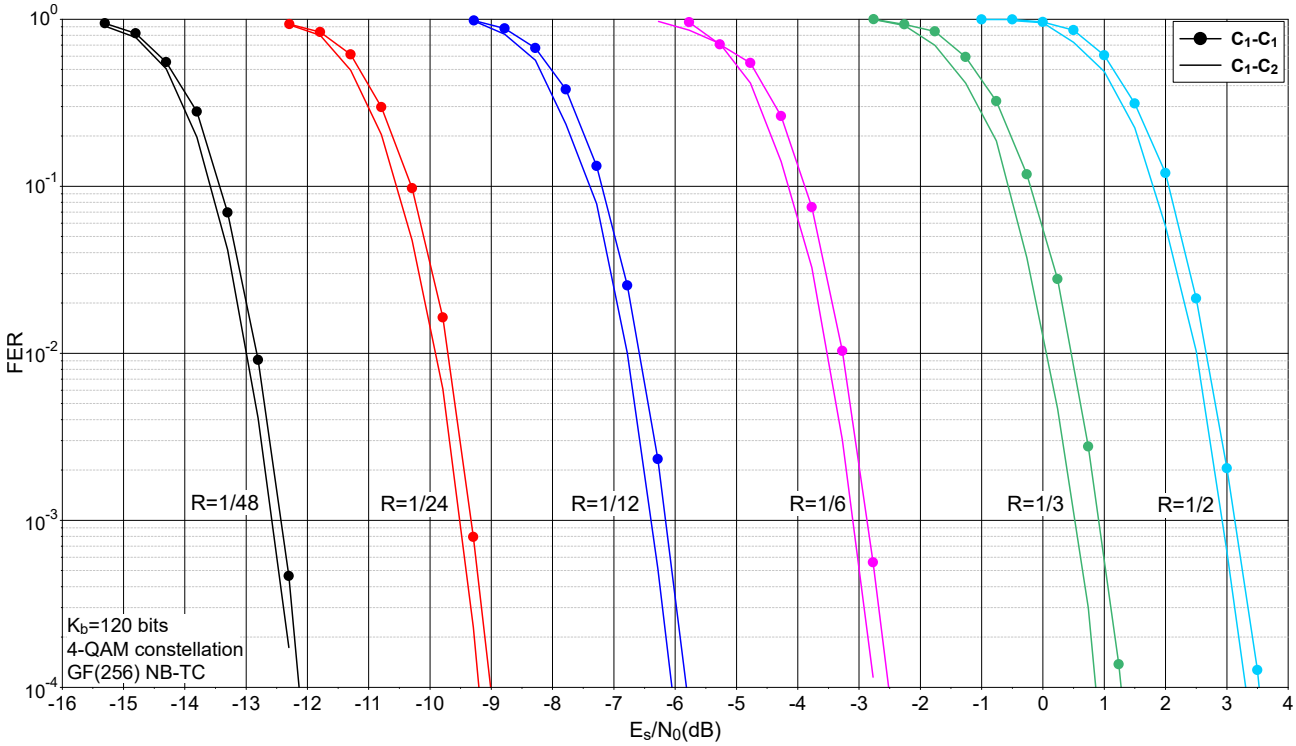


Figure 2.4: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(256), information frame size K=120 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

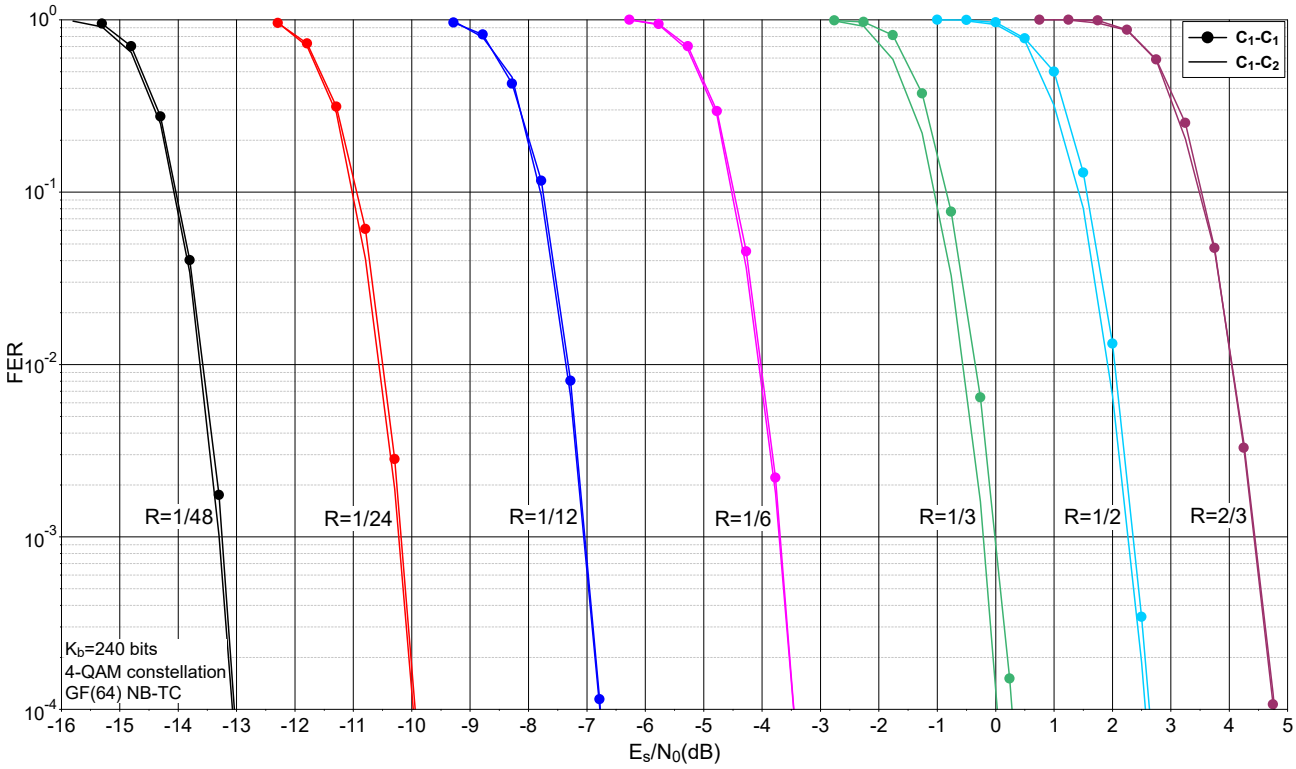


Figure 2.5: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(64), information frame size K=240 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

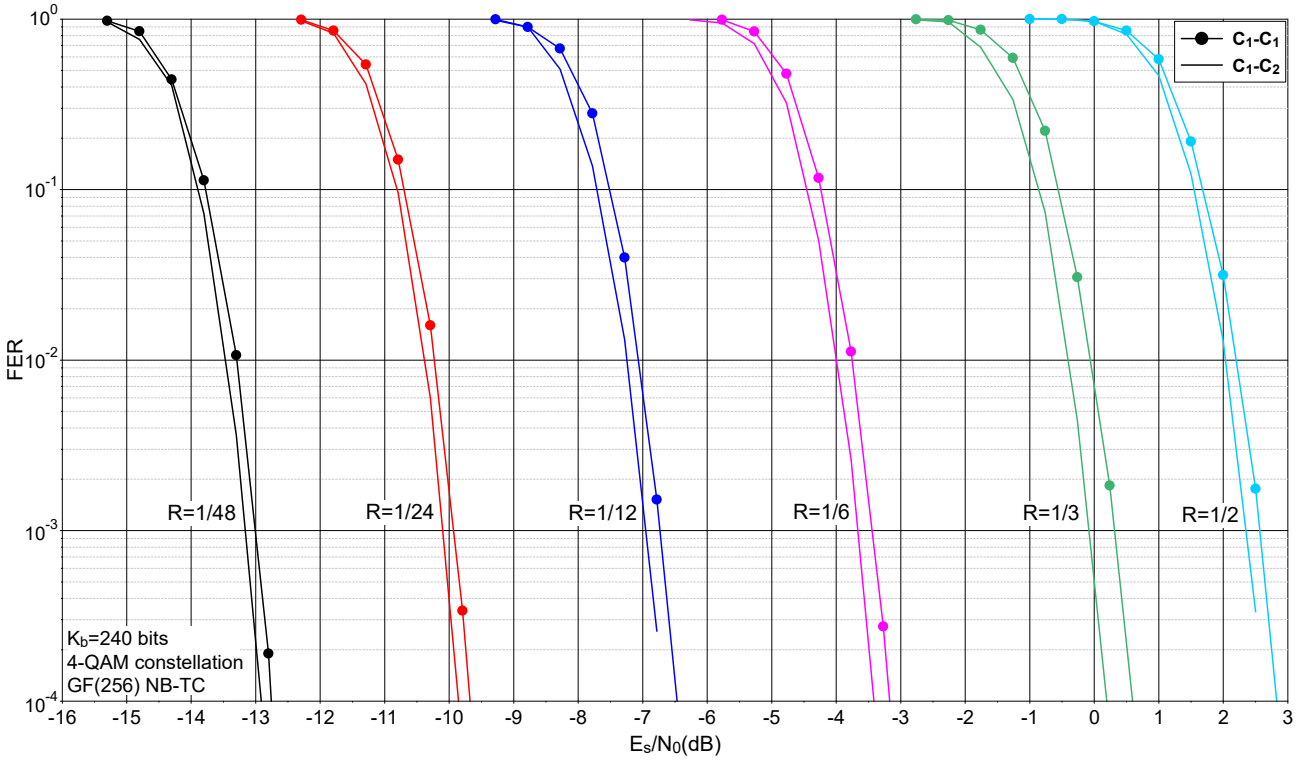


Figure 2.6: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(256), information frame size K=240 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

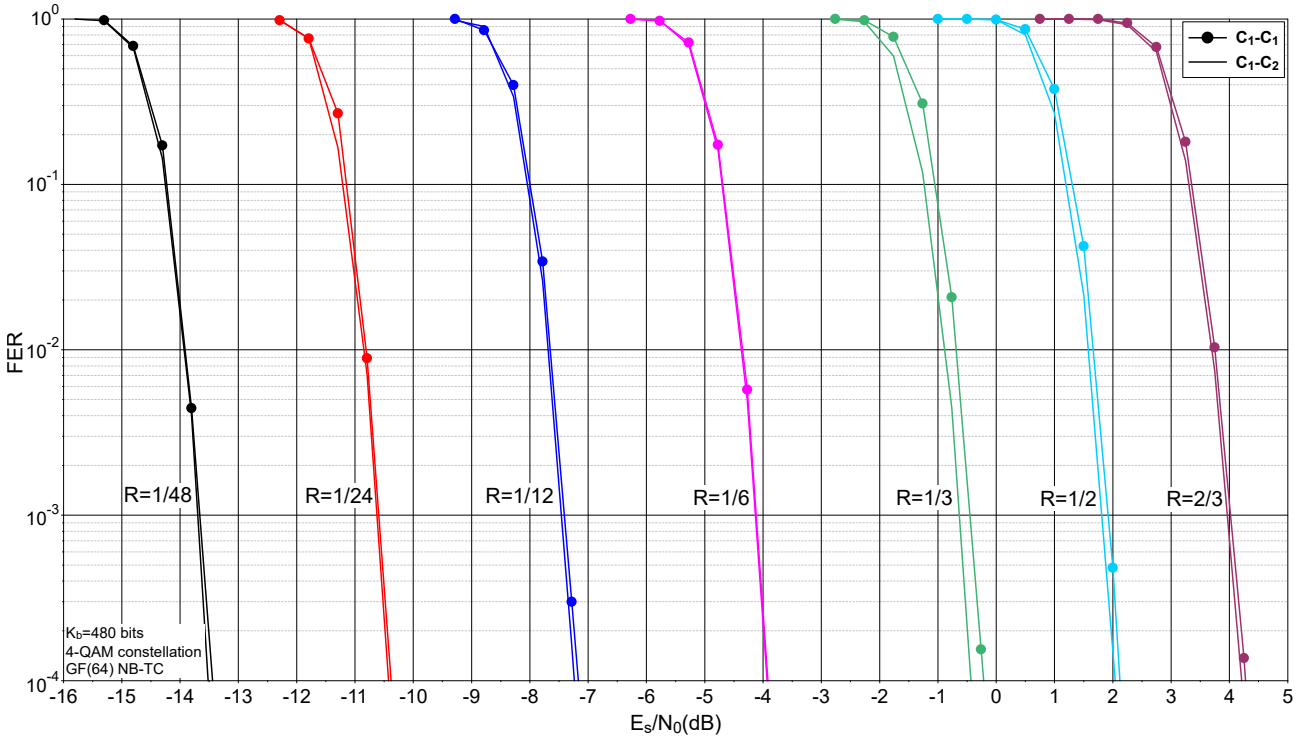


Figure 2.7: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(64), information frame size K=480 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

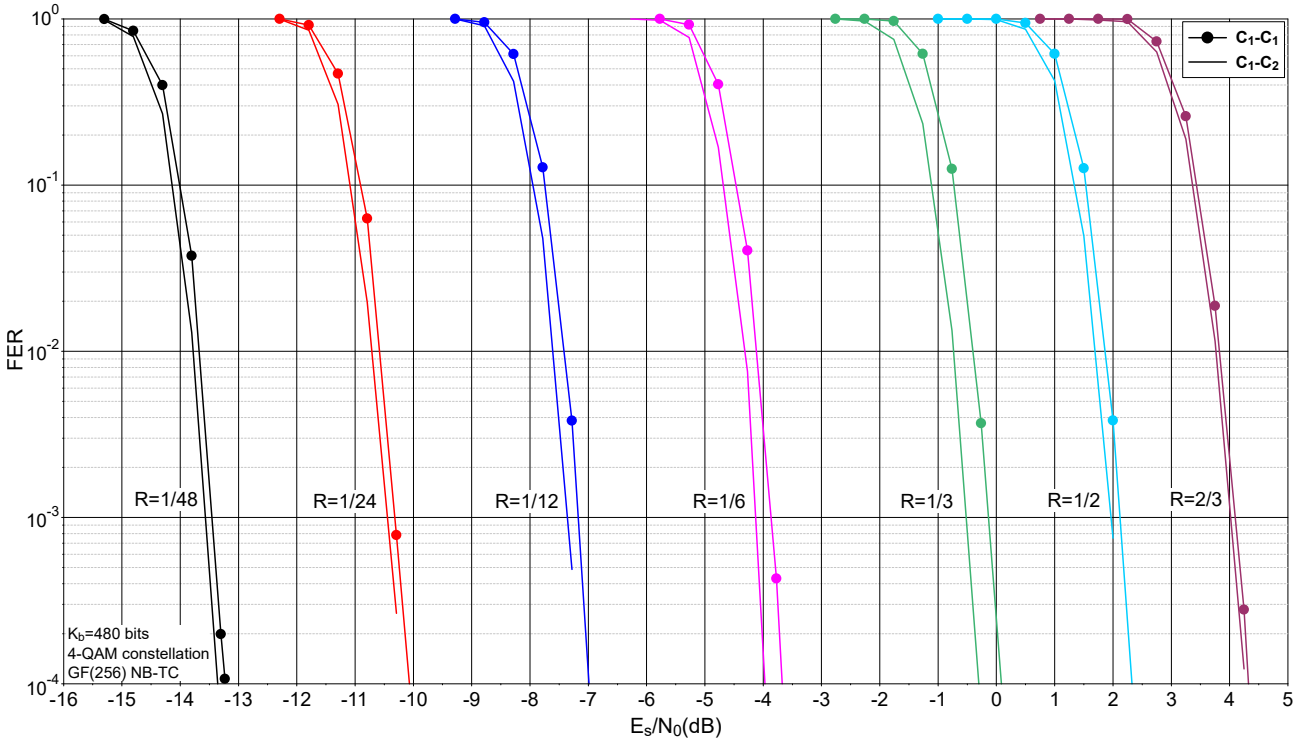


Figure 2.8: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(256), information frame size K=480 bits, code rates R=1/48, 1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

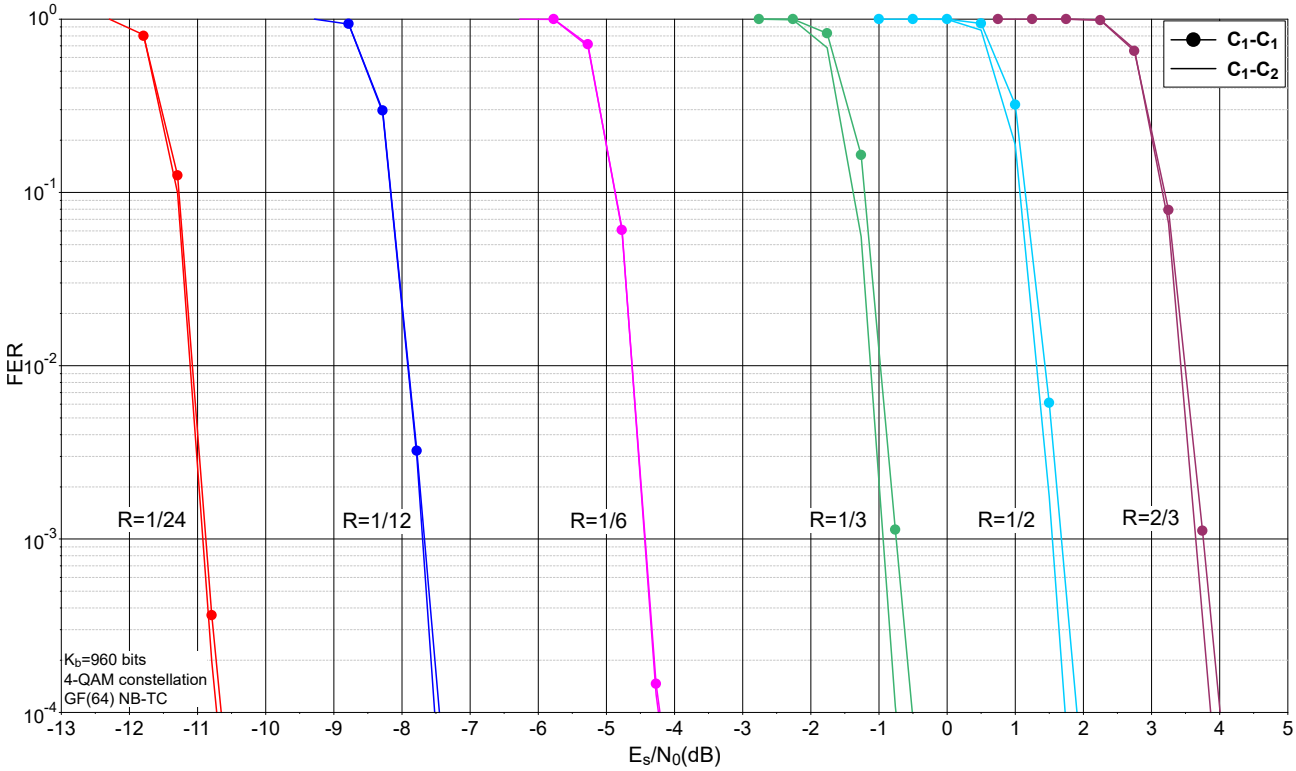


Figure 2.9: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(64), information frame size K=960 bits, code rates R=1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

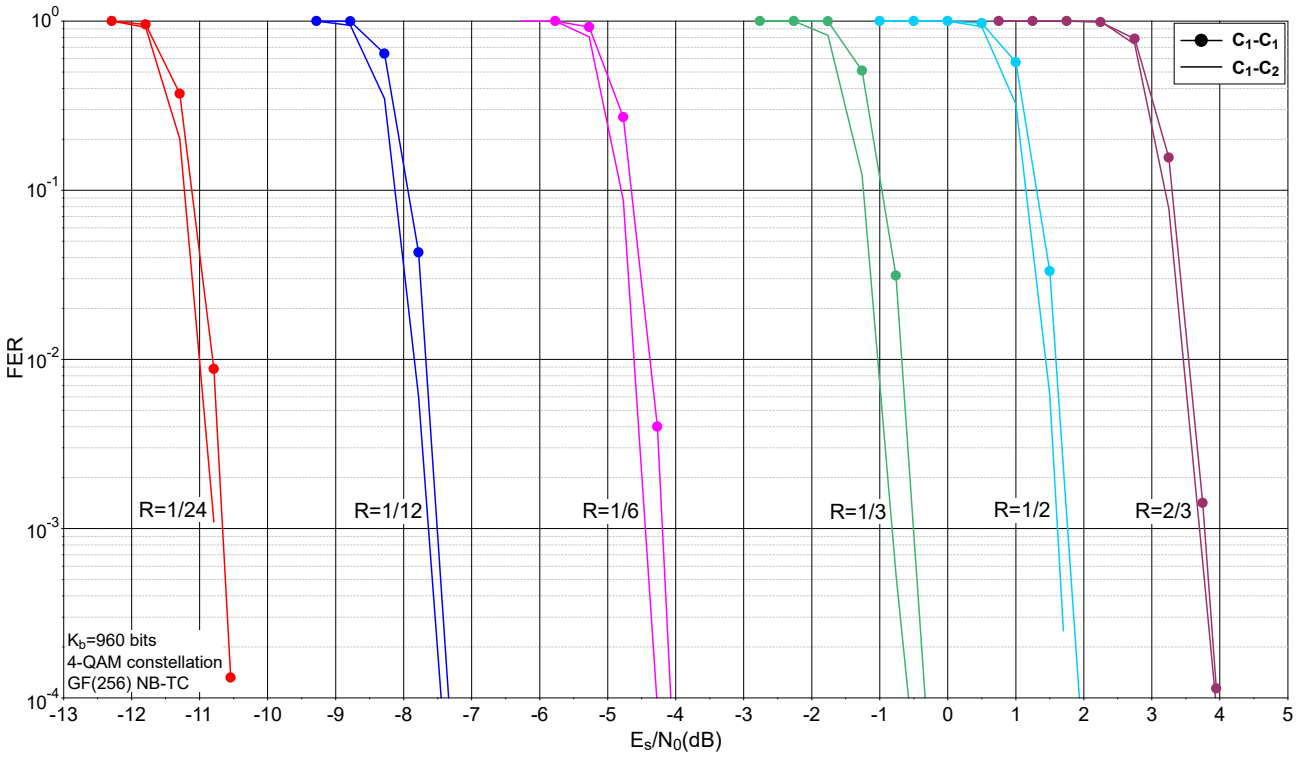


Figure 2.10: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(256), information frame size K=960 bits, code rates R=1/24, 1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

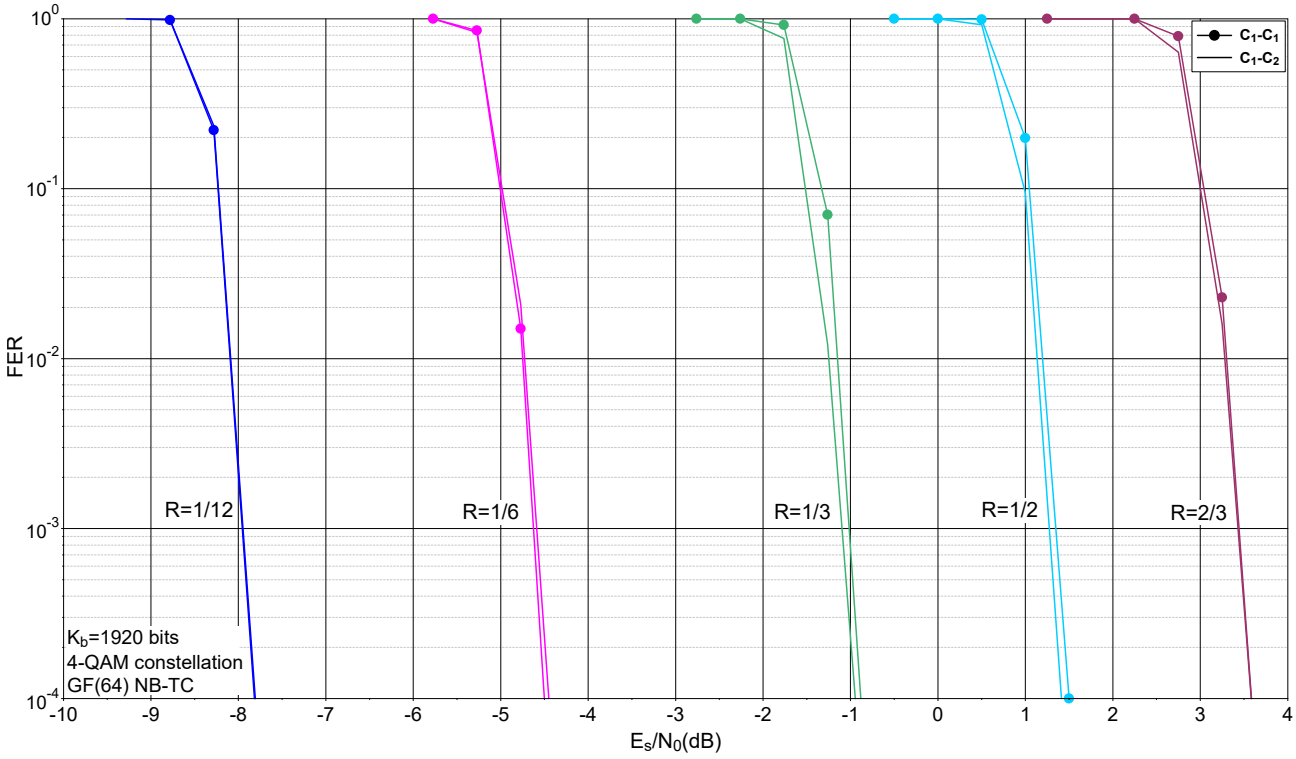


Figure 2.11: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(64), information frame size K=1920 bits, code rates R=1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

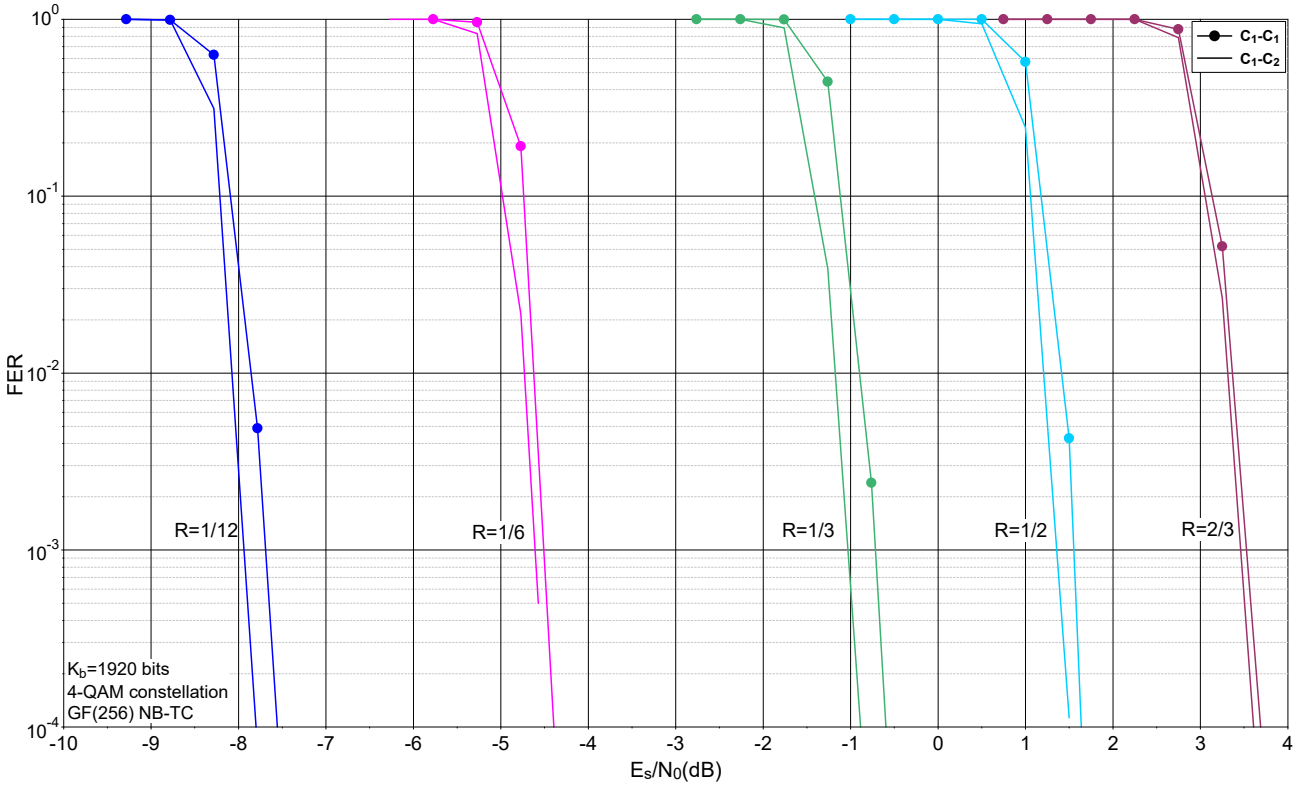


Figure 2.12: FER performance comparison between the NB-TC with the C_1 code and the one with a combination of C_1 and C_2. GF(256), information frame size K=1920 bits, code rates R=1/12, 1/6, 1/3, 1/2, 2/3 and AWGN channel. Scaled Max-Log MAP decoding and 8 turbo iterations.

2.4 Conclusion/summary

We have designed NB-TCs by defining constituent codes based on an improved accumulator structure proposed in our recent work. Moreover, previously proposed best-in class interleaver design for binary codes was extended to be used for NB codes. It was shown to improve performance while providing regularity, helpful for a simplified implementation. Periodic puncturing was introduced for rate adaptation coupled with a procedure to provide code rate flexibility and incremental redundancy down to one symbol without a penalizing effect on performance. Being able to target extremely low error rates without error floors, a compromise in the choice of the component codes was made to favor performance at low SNRs. Designs were made for GF(64) and for GF(256) for all the frame sizes and code rates agreed upon in D1.1. They target an association with a 4-QAM constellation. In a second phase, designs for an association with a CCSK modulation will be made, in addition to the exploration of parameters capable of further improving performance at low error rates.

3 Design of non-binary LDPC codes

3.1 Introduction

In this section we provide simulation results of NB-LDPC codes on an NB-LDPC decoder, with and without CCSK modulation. State of the art of NB-LDPC code construction is presented followed by a description of the construction method and parameters of the code used in the simulations. Finally the simulation results are provided.

3.2 Design method/procedure

3.2.1 Presentation of NB-LDPC codes

Linear bloc codes are a class of error-correcting codes widely used. It is defined as a linear injective mapping between $F^K \rightarrow F^N$, where $N > K$ and F is a finite cardinality set of size q with usual additive and multiplicative properties (known as Galois Field GF(q)). When $q = 2$, the code is said to be a binary code over GF(2) = 0,1. When $q > 2$, the code is said to be "non-binary". We consider only GF(q) sets with $q \geq 64$ and q a power of 2, i.e., $q = 2^m$ with $m \geq 6$. The mapping is performed through a $N \times K$ generator matrix $\boldsymbol{G}$ associated to a vector $\boldsymbol{u}$ of size $K \times 1$ and a vector $\boldsymbol{c}$ of size $N \times 1$ given by $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{u}$. The set of the q^K vectors thus generated constitutes the code $\mathcal{C}$. For each code, there is a $(N - K) \times N$ matrix $\boldsymbol{H}$ with elements in GF(q) so that $\boldsymbol{c} \in \mathcal{C} \iff \boldsymbol{H}\boldsymbol{c}^{\boldsymbol{T}} = 0$. The matrix $\boldsymbol{H}$ is called the parity check matrix. The ratio $R = K/N$ is called the coding rate of the code. Shannon's theory states that if the code rate is smaller than the channel capacity, then reliable transmissions are achievable.

A "good" linear code should have several properties. It should have a high enough minimum Hamming distance d_H (minimum number of non-null elements of a non-zero codeword). It should also be easy to encode, i.e., the generator matrix $\boldsymbol{G}$ should be sparse and/or structured. More importantly, the code should be associated with an efficient decoding algorithm and the code should also exhibit no weaknesses for the selected decoding algorithm.

3.2.2 NB LDPC code construction state of the art

When the proportion of non-null coefficients of the parity check matrix is low, the code is called a Low Density Parity Check (LDPC) code. In case of code over GF(q), with $q \geq 64$, it is shown that regular code with a constant variable degree d_v of $d_v = 2$ and a constant check node degree d_c can give a good code of rate $R \geq 1–2/d_c$ [10–12]. A method to construct good codes has been proposed in [13] and enhanced in [14] [15]. It is based on a 3-step process: 1) selection of a proto-matrix with high girth (i.e. the minimum size of the cycle in the graph)[13] and/or good decoding threshold [12] under EXIT chart analysis, 2) expansion of the proto-graph to the final graph by maximizing its girth, and 3) affectation of the GF symbols. The expansion process involves replacing each non-null coefficient of the regular $(d_v, d_c) = (2, d_c)$ protomatrix by a permutation matrix of size (P, P) and each 0 element of the prototype matrix by the (P, P) zero matrix. Since a permutation matrix contains exactly one item per row and per column, the expanded matrix is also a regular $(2, d_c)$ matrix. When the permutation set is restricted to a cyclic rotation, the code obtained is called Quasi-Cyclic LDPC [16]. In this case, a cycle in the prototype matrix is suppressed in the expanded matrix if and only if the rotation factors found along the cycle verify a linear equation [16]. Some papers propose more complex permutations, such as affine permutations for example [17]. Once the expanded matrix is generated, the affectation of GF coefficients should be conducted on the non-null position of the expanded parity check matrix. The set of coefficients in each row should be selected so that locally, the constructed code shows good performance [13] [18]. Moreover, the set of GF coefficients among small cycles should also verify a

non-null condition to suppress low weight codewords in the final code [13]. In [12], the authors propose to perform the affectation task of GF coefficients before the operation of matrix expansion. In other words, all the GF belonging to the same expanded matrix are equal. Although reducing drastically the area of research, the authors show that this constraint does not affect the quality of the constructed code, at least for the size and code rate they have constructed. Finally, in[19], a technique to match Quadratic Amplitude Modulation (QAM) labelling with GF coefficients of a parity check equation is presented. Considering the local code generated by a given parity check equation, the optimized labelling allows to minimize the number of couple of codewords with minimum Euclidian distance, thus improving the overall decoding performance of the decoder. Finally, it is worth mentioning that the cluster NB-LDPC code has a minimum distance that can increase linearly with the size of the code under some conditions [20] [21]. This opens the way for constructing strong long-sized codes in practice.

3.2.3 Description of the code

The NB codes are Regular code with code rate $R = K/N = 1/3$, $d_v = 2$ and $d_c = 3$. For GF(64), the information size are K=120, 240, 480, 960 and 1920 bits. For GF(256), the information size are K=120 and 240 bits. The alist format of the codes can be found in the NB-LDPC web page `http://www-labsticc.univ-ubs.fr/nb_ldpc/`. The matrices on GF(64) have been designed by Alireza Tasdighi, and the matrices on GF(256) have been designed by Titouan Gendron.

3.2.4 NB code construction

The general way of constructing NB-codes with $d_v = 2$ is based on four steps. **Step 1:** use an existing high girth, simple and regular graph such as cage or cubic graph. **Step 2:** each edge in the graph is replaced by a Variable Node of $d_v = 2$. The resultant graph is a bipartite Tanner graph with $d_v = 2$. **Step 3:** expend the graph by a Lifting Factor. **Step 4:** find the cyclic rotations and GF coefficients among small cycles that verify a non-null conditions to increase the girth. Parameters of the GF(64) codes are given in Table 3.1. For each information size K is given the obtained Girth, the graph used as prototype matrix and the lifting factor. The cubic graph are obtained from `http://hog.grinvin.org/Cubic`

K	Girth	Cubic graph	Lifting Factor
120	16	Girth 8, 40 Vertices	1
240	20	Girth 4, 8 Vertices	10
480	20	Girth 4, 8 Vertices	20
960	24	Girth 4, 8 Vertices	40
1920	28	Girth 6, 20 Vertices	32

Table 3.1: GF(64) NB code parameters

The parameters of the GF(256) codes are given in Table 3.2.

K	Girth	Cage graph	Lifting Factor
120	12	n=9	5
240	16	n=9	10

Table 3.2: GF(256) NB code parameters

Codes of different length and code rate are available on `http://www-labsticc.univ-ubs.fr/nb_ldpc/` with simulation results.

3.3 Parameters of the simulations

3.3.1 NB-LDPC decoder

The NB-LDPC decoder uses Extended Min-Sum algorithm [22] and Forward Backward Check Node (CN) implementation [23]. With $d_c = 3$, a CN can be efficiently implemented from 3 Elementary Check Nodes (ECN) [23] processed in parallel. Furthermore, the code with $d_v = 2$ allows an efficient overlapped processing for both check node unit and variable node unit [24]. The c code of the decoder used for simulation can be found in `https://github.com/Lab-STICC-UBS/NB_LDPC_FB_public`. For the simulations, the maximum number of iteration is set at $it_{max} = 30$, the parameter of the ECNs are for GF(64): $n_m = 20$, $n_{op} = 25$ and offset=1.0, and for GF(256): $n_m = 80$, $n_{op} = 90$ and offset=1.0 .

3.3.2 CCSK modulation

The Cyclic Code Shift Keying (CCSK) modulation use a Pseudo-Noise sequence of size q generated using an extended m-sequence [25]. The feedback polynomial $g(x)$ of the m-sequence is $g(x) = x^6 + x^5 + x^4 + x + 1$ for $q = 64$ and $g(x) = x^8 + x^4 + x^3 + x^2 + 1$ for $q = 256$. The CCSK modulation maps an element $k \in GF(q)$ to the sequence $\mathbf{P}_k$ defined as the circular shift of $\mathbf{P}_0$ by k positions.

3.4 Simulation results

Parameter of the decoder, the code and CCSK modulation are given in Sections 3.3. We provide first simulation results of the NB decoder with GF(64) and GF(256), then the simulation results of the NB decoder combined with CCSK modulation.

3.4.1 Simulation results of the NB decoder

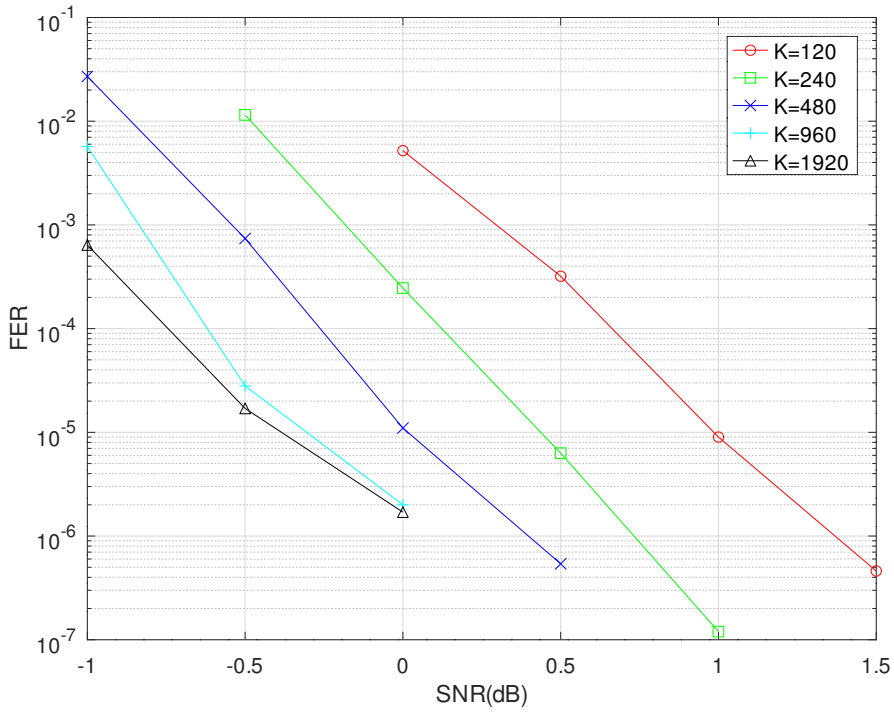


Figure 3.1: FER performance comparison with GF(64)

Fig. 3.1 shows simulation results on GF(64) for K=120, 240, 980 and 1920 bits.
Fig. 3.2 shows simulation results on GF(256) for K=120 and 240 bits.

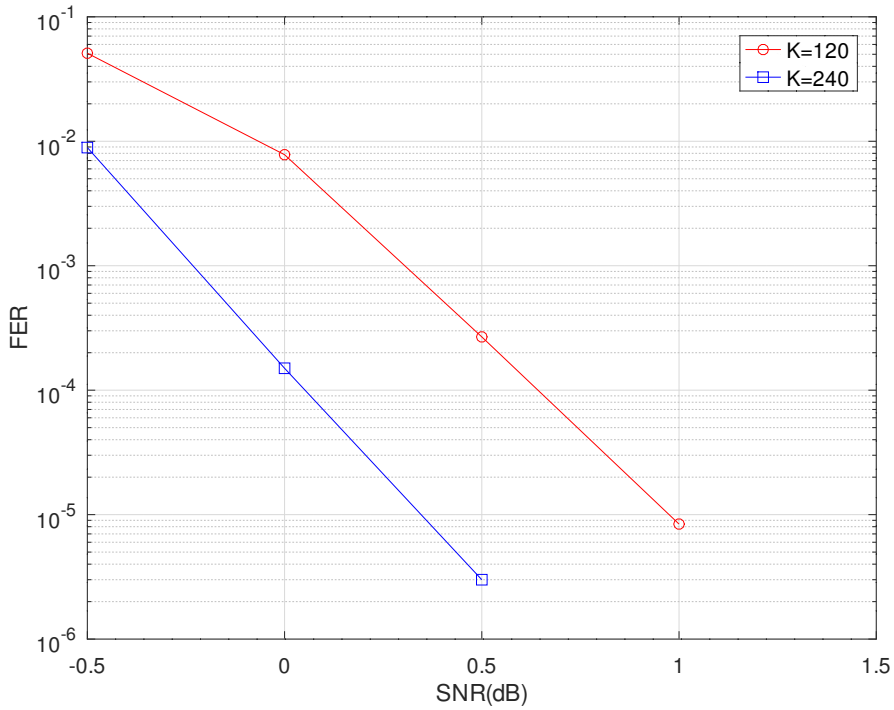


Figure 3.2: FER performance comparison with GF(256)

3.4.2 Simulation results of the NB decoder combined with CCSK modulation

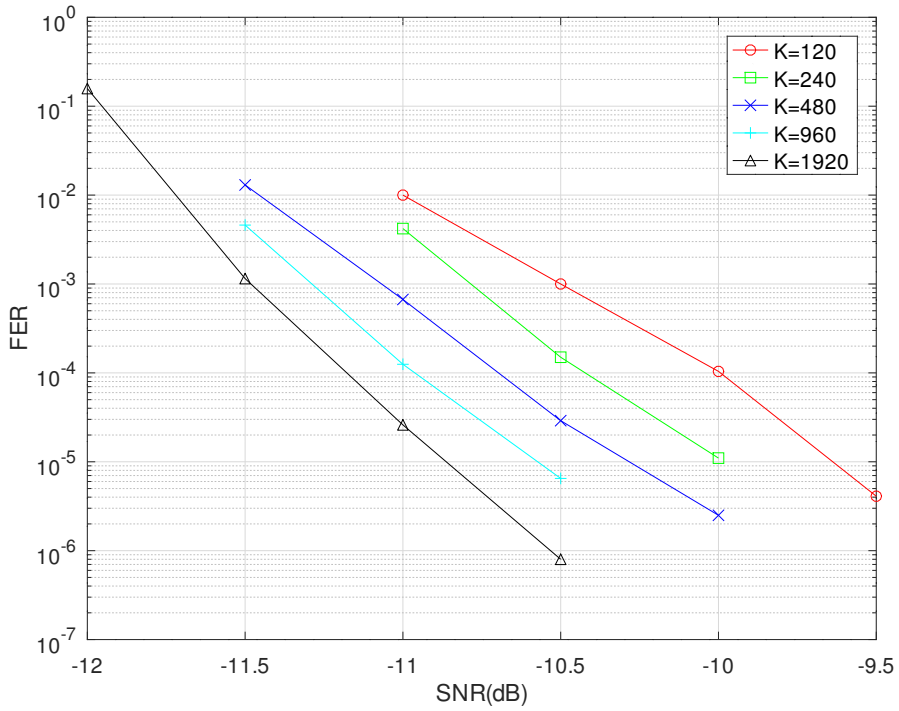


Figure 3.3: FER performance comparison with GF(64) combined with CCSK

Fig. 3.3 shows on GF(64) for K=120, 240, 980 and 1920 bits combined with CCSK modulation.
Fig. 3.4 shows on GF(256) for K=120 and 240 bits combined with CCSK modulation.

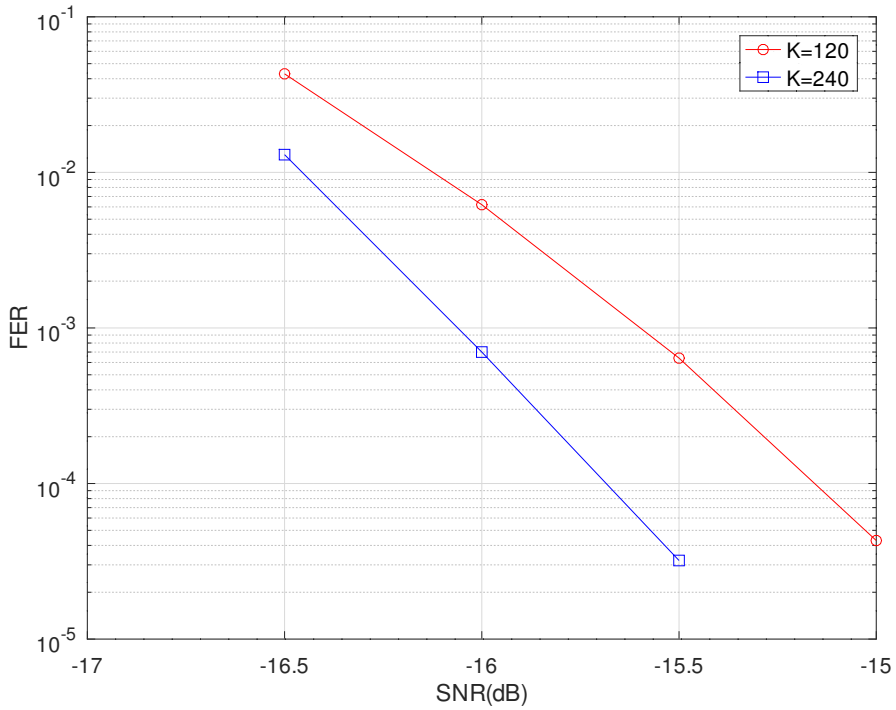


Figure 3.4: FER performance comparison with GF(256) combined with CCSK

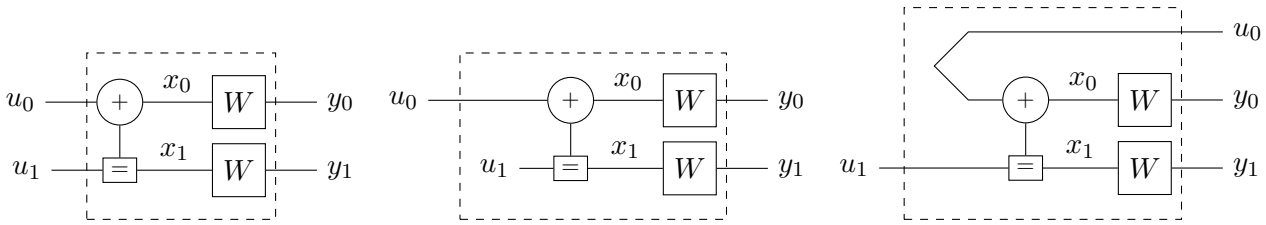
3.5 Conclusion/summary

In this section we provide simulation results of NB-LDPC codes on an NB-LDPC decoder, with and without CCSK modulation. The NB-LDPC codes are available in `http://www-labsticc.univ-ubs.fr/nb_ldpc/`. The c code of the NB decoder is available in `https://github.com/Lab-STICC-UBS/NB_LDPC_FB_public`. The c code for the CCCSK modulation is not public yet.

4 Design of non-binary Polar codes

4.1 Introduction

Channel polarization was initially introduced by Arıkan [26], as a mean for constructing codes achieving the symmetric capacity of any binary-input discrete memoryless channel. The construction relies on a *channel combining and splitting* procedure, as illustrated at Fig. 4.1. First, two instances of a transmission channel are combined by entangling their inputs through a linear transformation. The combined channel is then split into two *virtual channels*, defined by randomizing one of the two inputs of the combined channel. For the *bad channel*, the value of the randomized input is unknown to the receiver, while for the *good channel*, the value of the randomized input is made known to the receiver.



(a) Combined channel (b) Bad channel (c) Good channel

Figure 4.1: Channel combining and splitting. (a) The combined channel uses two instances of the transmission channel W, whose inputs are given by the linear transformation $x_0 = u_0 \oplus u_1, x_1 = u_1$. (b) For the bad channel, the second input of the combined channel is set to a random value, unknown to the receiver. (c) For the good channel, the first input of the combined channel is set to a random value, which is also made known to the receiver.

Applied recursively, the above procedure synthesize a set of N virtual channels from N instances of the transmission channel, with $N = 2^n$, where n is the depth (number of steps) of the recursion. The synthesized channels show a polarization effect, in the sense that they tend to become either perfect (noiseless) or useless (completely noisy). In practice, the channel polarization phenomenon can be effectively exploited by feeding the perfect synthesized channels with information bits, and freezing the useless ones to zero (or to any values known to both transmitter and receiver). A successive cancellation (SC) decoding algorithm can be applied at the receiver, to restore the information bits.

Central to the above construction is the linear transformation, also known as *kernel transformation* or simply *kernel*, used to combine two instances of a same transmission channel. The original kernel used by Arikan for binary-input channels, illustrated at Fig. 4.1(a), is defined by:

$$\begin{cases} x_0 &= u_0 \oplus u_1 \\ x_1 &= u_1 \end{cases} \tag{4.1}$$

where $\oplus$ denotes the sum modulo 2 (XOR) operation. This kernel is readily generalizable to channels with non-binary input alphabets, provided that the non-binary alphabet is endowed with an additive group structure[1] $\oplus$. If the size of the alphabet is a prime number, polarization happens in two levels, similarly to the binary alphabet case, with synthesized virtual channels polarizing to either useless or perfect channels [27]. However, in the general case, channel polarization may happen in several levels, with synthesized virtual channels polarizing to either useless, perfect or *intermediate* channels.

[1]Note that it is always possible to endow a non-binary alphabet with an additive group structure $\oplus$. One possibility is to label the alphabet symbols from 0 to $q-1$, where q is the size of the alphabet, and then to take $\oplus$ to be the sum modulo q. Of course, there may be other possibilities not equivalent (that is, *not isomorphic*) to this one. If q is a power of 2, a different possibility is to take $\oplus$ to be the bitwise-XOR operation.

The approach proposed in [27] in case the input alphabet size is not a prime, is to decompose W into subchannels of prime input alphabet sizes, and then apply a multi-level code construction technique, by polarizing each subchannel separately. A different approach was proposed in [28], where it has been shown that intermediate channels can simply be dealt with, by *partly freezing* their inputs. Precisely, the input of an intermediate channel is constrained to a subset[2] only of the non-binary alphabet, thus requiring only a slight modification of the encoding and decoding rules (as usual, perfect channels are feed with information symbols, and useless ones frozen).

There are two known approaches to polarize non-binary channels, with arbitrary input alphabet sizes, into either perfect or useless channels only (two-level polarization). The first one relies on using *higher-dimension* non-binary kernels, that is, kernels of size $\ell \times \ell$, with $\ell > 2$ [29–31]. Such an approach is characterized by an increased complexity, due to both the size of the non-binary alphabet, and the higher kernel dimension. A different approach, proposed in [27], is to use a randomized construction, based on the original kernel proposed by Arikan. Precisely, the kernel transformation, is defined by:

$$\begin{cases} x_0 &= u_0 \oplus u_1 \\ x_1 &= \pi(u_1) \end{cases} \tag{4.2}$$

where π is a random permutation of the non-binary alphabet. A random coding argument is then used to prove the tow-level polarization of the non-binary channel. It essentially states that for a random choice of permutations throughout the recursive channel combining and splitting procedure, the synthesized virtual channels polarize to either useless or perfect channels. In this case, the *polar code construction* encompasses the *choice* of both channel combining permutations and virtual channels used to transmit information symbols. While a random choice of the channel combining permutations is good enough, it might not be optimal[3]. Thus, optimizing the channel combining permutations should allow accelerating the speed of polarization of the synthesized virtual channels. It is worth being noticing that once the code is constructed, randomness does no longer exist, and the complexity of polar code encoding and decoding is essentially the same as for the Arikan's kernel.

The non-binary polar codes considered in this work are based on the randomized construction described above. However, we consider non-binary polar codes defined on Galois fields (GF), and rather than random GF permutations, we consider linear permutations defined by the multiplication with a non-zero GF element. Precisely,

- We consider a non-binary channel W with input alphabet of size $q = 2^p$, for some $p > 0$, and thus identify[4] the channel input alphabet with the Galois field with q elements, denoted $\mathrm{GF}(q)$. We denote by $\oplus$ and $\cdot$ the additive and multiplicative operations in $\mathrm{GF}(q)$.
- Kernel transformations, used within the recursive channel combining and spiting procedure, are defined by:

$$\begin{cases} x_0 &= u_0 \oplus u_1 \\ x_1 &= h \cdot u_1 \end{cases} \tag{4.3}$$

  with $h \in \mathrm{GF}(q) \setminus \{0\}$. In the sequel, h will be referred to as *kernel coefficient*. The kernel transformation is further illustrated at Fig. 4.2

It is worth noticing that the result in [27] states that polarization happens for random permutations of the non-binary alphabet. However, using arguments similar to those in [27], it can be proven that polarization also happens when the set of permutations is restricted to those defined by non-zero elements of $\mathrm{GF}(q)$.

The design methodology, aimed at optimizing the choice of the kernel coefficients is presented in Section 4.2. Non-binary polar decoding is also briefly presented in this section, and we propose an

[2] Actually, a subgroup, *i.e.*, a subset closed under the additive law $\oplus$.

[3] Similarly, for a non-binary LDPC code, a random choice of a non-binary coefficients ("edge permutations") is in general a good one, but not optimal.

[4] Such an identification can be made by using any one-to-one mapping between the channel input alphabet and $\mathrm{GF}(q)$.

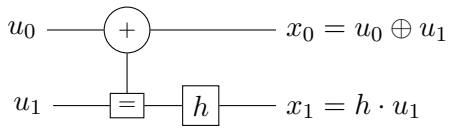


Figure 4.2: GF kernel, with h a non-zero elements of GF(q).

extension of the SC-List decoder to the case of non-binary polar codes. Different puncturing strategies are also discussed, supporting the choice of a simple but effective puncturing strategy for non-binary codes. Numerical results are presented in Section 4.3, for the additive white Gaussian noise (AWGN) channel, with either binary (± 1) or cyclic codes shift keying (CCSK) modulated inputs. Note that in case of binary inputs, the polarized non-binary channel W is defined by aggregating p instances of the binary-input channel. The binary-input AWGN channel serves as a reference model and comparison with the other families of codes investigated in the project. In case of CCSK modulated inputs, the transmission channel is natively non-binary, and we consider polar codes defined on GF(q), where $q = 2^p$ is the size of the CCSK symbols.

4.2 Design method/procedure

4.2.1 Optimization of the kernel coefficients

As mentioned in the Introduction, the optimization of the kernel coefficients is aimed at accelerating the speed of polarization of the synthesized virtual channels. There are three parameters that may be used to describe the polarization process: the mutual information, the Bhattacharyya parameter, and the error probability of the synthesized virtual channels. The former approaches 0 (respectively[5], 1) if and only if the latter two approach 1 (respectively, 0). Any of these parameters may be used within the proposed optimization procedure, and for the moment we shall simply use *polarizing parameter* to refer to any of them. To accelerate the speed of polarization, we choose the kernel coefficients so as to maximize the difference between the polarizing parameters of the bad and good channels synthesized by the channel combining and splitting procedure.

The optimization procedure is illustrated at Fig. 4.3, for a polar code of length $N = 8$, corresponding to $n = 3$ *polarization steps*. The original non-binary channel is denoted by W. We denote by $W^{(0)}$ and $W^{(1)}$ the bad and good channels, respectively, after one step of polarization. Then, for $n > 0$, we define recursively

$$W^{(i_1 \dots i_{n-1} i_n)} := \left(W^{(i_1 \dots i_{n-1})}\right)^{(i_n)}, \quad \forall (i_1 \dots i_n) \in \{0,1\}^n \tag{4.4}$$

In Fig. 4.3, we have indicated on each horizontal wire the virtual channel $W^{(i_1 i_2 \cdots)}$ "seen" by the corresponding symbol throughout the polarization process. All the kernels on the first (right-most) polarization step combine two copies of the W channel. Therefore, only one coefficient needs to be optimized, denoted by h_0. We define h_0 as

$$h_0 := \underset{h \in \mathrm{GF}(q)}{\operatorname{argmax}} \left| P^{(0)}(h) - P^{(1)}(h) \right|, \tag{4.5}$$

where $P^{(0)}(h)$ and $P^{(1)}(h)$ denote the polarizing parameters of $W^{(0)}$ and $W^{(1)}$ channels, respectively, assuming that the channel combining coefficient is equal to h. We numerically estimate the values of $P^{(0)}(h)$ and $P^{(1)}(h)$, for all $h \in \mathrm{GF}(q) \setminus \{0\}$, based on Monte Carlo simulation (see also Section 4.2.3).

Once the value of h_0 is determined, we can optimize the kernel coefficients for the second (middle) polarization step. There are two different types of kernels on the second polarization step, combining either two copies of $W^{(0)}$, or two copies of $W^{(1)}$. Therefore, two coefficients need to be optimized,

[5]We assume here that the mutual information is normalized (expressed in terms of symbols per channel use), thus taking values between 0 and 1.

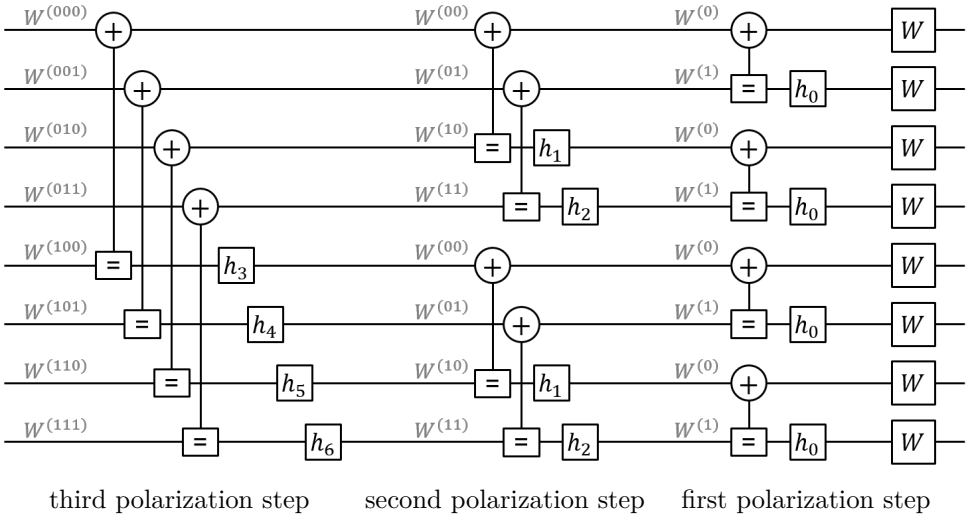


Figure 4.3: Non-binary polar code of length $N = 8$, corresponding to $n = 3$ steps of polarization

denoted by h_1 and h_2 in Fig. 4.3. Hence, we define

$$h_1 := \underset{h \in \mathrm{GF}(q)}{\operatorname{argmax}} \left| P^{(00)}(h) - P^{(10)}(h) \right|, \tag{4.6}$$

$$h_2 := \underset{h \in \mathrm{GF}(q)}{\operatorname{argmax}} \left| P^{(10)}(h) - P^{(11)}(h) \right|, \tag{4.7}$$

where $P^{(i_1 i_2)}(h)$ denotes the polarizing parameters of the $W^{(i_1 i_2)}$ channel, assuming the channel combining coefficient on the second polarization step is equal to h. The value of $P^{(i_1 i_2)}(h)$ is again estimated numerically through Monte Carlo simulation. Then the optimization process continue recursively, until the desired number of polarization steps is reached.

4.2.2 Non-binary polar decoding

We first consider the decoding of a non-binary kernel, which is illustrated at Fig. 4.4. As before, let $u_0, u_1 \in \mathrm{GF}(q)$ denote the kernel inputs, and $x_0, x_1 \in \mathrm{GF}(q)$ denote the kernel outputs. Decoding operates in the opposite direction, *i.e.*, it takes as inputs $\mathbf{P}_X^{(0)}$ and $\mathbf{P}_X^{(1)}$, the probability mass functions (PMFs) of x_0 and x_1, respectively, and outputs $\mathbf{P}_U^{(0)}$ and $\mathbf{P}_U^{(1)}$, the PMFs of u_0 and u_1, respectively. It can be easily seen that $\mathbf{P}_U^{(0)}$ and $\mathbf{P}_U^{(1)}$ can be computed from $\mathbf{P}_X^{(0)}$ and $\mathbf{P}_X^{(1)}$, by the following formulas:

$$\mathbf{P}_U^{(0)}(u) = \sum_{u' \in \mathrm{GF}(q)} \mathbf{P}_X^{(0)}(u \oplus u') \mathbf{P}_X^{(1)}(h \cdot u') \tag{4.8}$$

$$\mathbf{P}_U^{(1)}(u) = \eta \mathbf{P}_X^{(0)}(u_0 \oplus u) \mathbf{P}_X^{(1)}(h \cdot u), \tag{4.9}$$

where η is a normalization factor, such that $\sum_{u \in \mathrm{GF}(q)} \mathbf{P}_U^{(1)}(u) = 1$.

(a) Bad channel decoding

(b) Bad channel decoding

Figure 4.4: Decoding of bad and good virtual channels. For decoding the good channel, the decoder uses the knowledge of u_0 (genie-aided decoder), or an estimate of it, $\hat{u}_0$ (real-world decoder).

In equation (4.9), the computation of $\mathbf{P}_U^{(1)}(u)$ requires the knowledge of u_0. Such a decoder is referred to as *genie-aided*, and it is used at the code design stage. For a *real-world decoder*, used to decode a codeword transmitted over a noisy channel, equation (4.9) is replaced by

$$\mathbf{P}_U^{(1)}(u) = \eta \mathbf{P}_X^{(0)}(\hat{u}_0 \oplus u)\mathbf{P}_X^{(1)}(h \cdot u), \tag{4.10}$$

where either $\hat{u}_0 = u_0$ if the latter is known (frozen bad channel), or $\hat{u}_0 = \text{argmax}_{u \in \text{GF}(q)} \mathbf{P}_U^{(0)}(u)$ is the estimate of u_0, otherwise.

The successive cancellation (SC) decoder (either genie-aided or real-world) uses the above kernel decoding rules in a recursive manner, so as to propagate the PMFs of the transmitted symbols from the right-hand side (transmission channel side) to the left-hand side of the polar graph. There is one such a recursion for each $W^{(i_1 \dots i_n)}$ channel, $(i_1 \dots i_n) \in \{0,1\}^n$, following the recursive definition of the channel. Finally, $W^{(i_1 \dots i_n)}$ channels are then decoded successively: once a channel is decoded, a hard decision is taken on the corresponding input and reused for decoding the subsequent channels.

For short to moderate code-lengths, the "incomplete polarization" of the virtual channels may drastically penalize the error correction performance of the SC decoder. Here, "incomplete polarization" means that the error probability of some virtual channels carrying information symbols may not be sufficiently small, thus dominating the SC decoding error probability. Indeed, an error (wrong hard decision) on such a channel will make the SC decoding fail, since SC decoding only moves forward, and cannot reverse any of the taken decisions. To address this problem, an SC-List decoding has been proposed in [32] for binary polar codes. After decoding a virtual channel carrying an information bit, SC decoding is duplicated in two parallel decoding paths, corresponding to the two possible hard decisions. In order to avoid an exponentially growing complexity, the number of parallel decoding paths is limited to a chosen parameter L, and the L surviving paths are determined according to a *path metric*. Moreover, to advantageously exploit the potential of SC-List decoding, the concatenation of an outer Cyclic Redundancy Check (CRC) code has also been proposed in [32], to help identifying the correct message within the decoded list. Concatenated CRC-Polar codes under SC-List decoding have been shown to compete with other families of error correcting codes, such as Low Density Parity Check (LDPC) and Turbo codes.

For non-binary polar codes, we extended the SC-List decoding approach from [32] as follows. After decoding a virtual channel carrying an information symbol, SC decoding is replicated in q parallel decoding paths, corresponding to the q possible hard decisions. In order to avoid an exponentially growing complexity, the number of parallel decoding paths is again limited to a chosen parameter L. Note that L may be much smaller than q (as a matter of fact, simulation results from Section 4.3 show the decoding performance of the SC-List decoding with $L = 8$, for polar codes on GF($q = 64$) and GF($q = 256$)). The L surviving decoding paths are determined according to a path metric, essentially corresponding to the probability of the path being correct. We also consider a CRC-assisted SC-List decoder, by concatenating an outer CRC code to the non-binary Polar code. Details will be provided in Deliverable D1.2.b.

4.2.3 Choice of the polarizing parameter and code construction

In Section 4.2.1 we presented an optimization procedure aimed at accelerating the polarization speed, where the latter has been described in terms of some *polarizing parameter*. As mentioned in Section 4.2.1, the polarizing parameter may be any of the mutual information, the Bhattacharyya parameter, or the error probability of the synthesized virtual channels. Any of the above three parameters can be numerically estimated by Monte Carlo simulation, using a genie-aided SC decoder to determine the PMFs of the inputs to the virtual channels $W^{(i_1 \dots i_n)}$, $(i_1 \dots i_n) \in \{0,1\}^n$.

Since we are interested on the error rate performance of the constructed polar code, we take the polarizing parameter used within the optimization procedure (from Section 4.2.1) to be the error probability of the synthesized virtual channels. An efficient way to numerically estimate the error probability of the synthesized virtual channels is described below, where $u^{(i_1 \dots i_n)} \in \text{GF}(q)$ denotes the input of the $W^{(i_1 \dots i_n)}$ virtual channel, $(i_1 \dots i_n) \in \{0,1\}^n$.

1. Randomly generate a set of inputs $\left\{u^{(i_1 \dots i_n)} : (i_1 \dots i_n) \in \{0,1\}^n\right\}$, encode them, and transmit the obtained codeword over the non-binary channel.
2. Run the genie-aided SC decoder to determine the PMFs of the virtual channels' inputs $u^{(i_1 \dots i_n)}$, denoted by $\mathbf{P}_U^{(i_1 \dots i_n)}$.
3. Hence, the *one-run error probability* of the virtual channel $W^{(i_1 \dots i_n)}$ is given by

$$P_{\text{one-run}}^{(i_1 \dots i_n)} = 1 - \mathbf{P}_U^{(i_1 \dots i_n)}\left(u^{(i_1 \dots i_n)}\right) \tag{4.11}$$

↻ Repeating the steps 1–3 many times, the error probability of the virtual channel $W^{(i_1 \dots i_n)}$ is estimated by taking the average of the one-run error probability:

$$P^{(i_1 \dots i_n)} = \mathbf{E}\left[P_{\text{one-run}}^{(i_1 \dots i_n)}\right] \tag{4.12}$$

The above procedure is used recursively within the optimization procedure from Section 4.2.1, to optimize the kernel coefficients at the different polarization steps. Moreover, once the optimization procedure completed, we may use the $P^{(i_1 \dots i_n)}$ values to sort the virtual channels from the best (lowest error probability) to the worst (higher error probability) one, and then use the best channels[6] to transmit information symbols. This completes the polar code construction, as we have made a choice of the kernel coefficients, and determined the virtual channels to use for transmitting information symbols.

Moreover, when the polar code construction completes, we also get an estimate the SC decoding error probability, denoted $\overline{\text{FER}}_{\text{SC}}$. To simplify the notation, let us denote by $P^{(1)}, \dots, P^{(K)}$ the error probability of the K virtual channels carrying information symbols. Then, we have

$$\overline{\text{FER}}_{\text{SC}} := 1 - \prod_{k=1,\dots,K}\left(1 - P^{(k)}\right) \tag{4.13}$$

For binary polar codes, $\overline{\text{FER}}_{\text{SC}}$ is known to provide a tight upper-bound on the frame error rate (FER) performance of the SC decoder (which also explains the notation). In Section 4.3, we will show that the same is true for non-binary polar codes.

Finally, we note that the above polar code construction (encompassing the optimization of the kernel coefficients and the virtual channels sorting) is dependent on the channel, and is carried out for each noise level (*e.g.* signal-to-noise, SNR) value.

4.2.4 Punctured polar codes

Assuming a polar code of length $N = 2^n$, defined on $\text{GF}(q = 2^p)$, the binary code length is given by $N_{\text{bin}} = Np$. To accommodate different code-lengths, either puncturing or shortening techniques may be considered. Both techniques have been investigated in the literature for binary polar codes, but to the best of our knowledge neither one has been investigated for non-binary polar codes. It is worth noticing that the optimization of either the puncturing or shortening pattern is in general a difficult problem. Let us restrict the discussion to the puncturing technique (but the same holds for the shortening, since it is essentially the dual of the puncturing technique). To construct a punctured polar code (that is, determine the bad/good virtual channel, and in case of non-binary codes, the kernel coefficients) we need to know the puncturing pattern indicating which coded symbols are transmitted over the channel. In a sense, the code doesn't exist before the puncturing pattern is specified. A brute-force search algorithm, testing all the possible puncturing patterns, is not doable in practice (unless the number of punctured symbols is very small).

For binary polar codes, the two main approaches proposed in the literature are either (1) construct first the unpunctured polar code, and then find a puncturing pattern that works well for the constructed

[6] We use the K best channels, where $K < 2^n$ is the number of information symbols to transmit.

code [33], or (2) choose fist a puncturing pattern that seems reasonably good, and then construct the polar code for the chosen pattern [34]. One of the best and simplest approaches is actually the second. The approach proposed in [34], referred to as quasi-uniform puncturing (QUP), is to puncture the coded bits in order, starting from the first one.

The theoretical tools introduced in [35, 36] allow classifying puncturing patterns in equivalence classes, with equivalent patterns provably yielding punctured codes with the same error correction performance. Then a constructive solution has been given, able to find a unique representative for each equivalence class of puncturing patterns. This allows finding the optimal puncturing pattern for short polar codes ($N \leq 64$), while avoiding brute-force search. The method may be also used (with some simplifications) for longer codes, but it does not guarantee the optimality of the solution. However, the results in [35] show that the optimal (or "quasi-optimal", for longer codes) puncturing pattern provides only a marginal gain over the QUP method. Is spite of its simplicity, the QUP method remains one of the most effective.

In [36], it has also been shown that puncturing and shortening are dual problems, and that optimization strategies for puncturing patterns can be readily transposed to shortening patterns. Comparing the optimal puncturing and shortening solutions for a short polar code ($N = 64$), it has been observed that puncturing yields better results for coding rates less than $1/2$, while shortening yields better results for coding rates greater than $1/2$ (similar observations have also been made in [37]).

Although the above discussion holds for binary codes, it gives us some confidence that similar approaches may be used for puncturing and/or shortening non-binary polar codes. Accordingly, since QCSP project targets non-binary codes with low coding rates (typically $1/3$ and below), we use puncturing to match the desired code length, when needed. In this case, we use a puncturing strategy similar to the QUP method for binary codes, and consisting in puncturing the required number of coded symbols, in order, starting with the first one. The non-binary code is then constructed (encompassing the optimization of the kernel coefficients and the virtual channels sorting) according to both the transmission channel and the puncturing pattern.

4.3 Simulation results

In this section we present simulation results for the binary-input AWGN (BI-AWGN) channel, and for the AWGN channel with CCSK modulated inputs. The following assumptions will be made:

- We consider polar codes defined on a Galois field $\mathrm{GF}(q)$, with $q = 2^p$. The non-binary code-length is given by $N = 2^n$, where n is the number of polarization steps.
- We denote by K_{bin} the binary code dimension (number of information bits), and by N_{bin} the binary code length (number of coded bits). Hence, $N_{\text{bin}} = Np$, unless for punctured polar codes, for which $N_{\text{bin}} < Np$.
- For the BI-AWGN channel, the polarized non-binary channel is defined by aggregating p instances of the binary-input channel.
- In case of CCSK modulated inputs, we consider CCSK symbols of the same size q as the polar code's Galois field. The polarized non-binary channel corresponds to the transmission of one CCSK symbol.
- We consider real Gaussian noise, with noise variance σ^2. Reported decibel values of the SNR are defined by $\mathrm{SNR} = -10\log_{10}(\sigma^2)\,(\mathrm{dB})$.

4.3.1 AWGN channel with binary-inputs

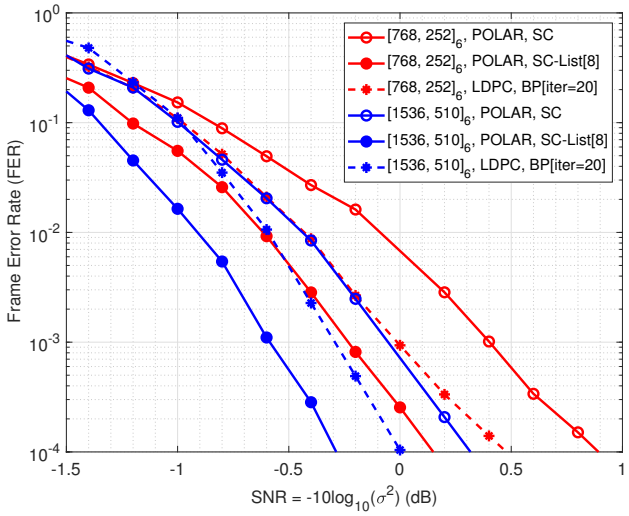
The BI-AWGN channel serves as a reference model and comparison with the other families of codes investigated in the project. We consider here only codes of rate $1/3$, but with different $(K_{\text{bin}}, N_{\text{bin}})$ parameters. In case that $N_{\text{bin}} < Np$ the codes are punctured by using a QUP pattern, as explained

in Section 4.2.4. Simulation results are given for both SC and SC-List decoders. The SC-List decoder is CRC-assisted (as explained in Section 4.2.2), and the list size is $L = 8$. We use a CRC of size 12 for codes defined on GF(64) and a CRC of size 8 for codes defined on GF(256).

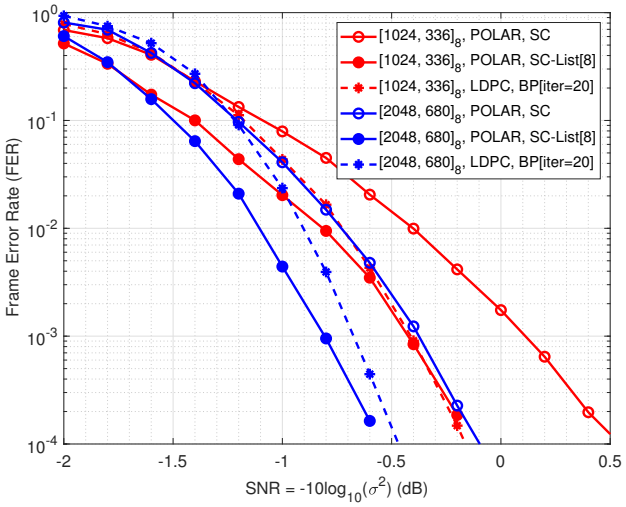
For comparison purposes, we have also constructed non-binary LDPC codes with the same $(K_{\text{bin}}, N_{\text{bin}})$ and p (Galois field) parameters. The decoding performance of the LDPC codes is obtained under Belief-Propagation, using 20 decoding iterations.

The frame error rate (FER) performance of unpunctured polar codes is shown at Fig. 4.5. The legend shows the $[N_{\text{bin}}, K_{\text{bin}}]_p$ parameters, the code type (polar or LDPC), and the decoding algorithm. Each color corresponds to the same $[N_{\text{bin}}, K_{\text{bin}}]_p$ parameters. Solid curves show the decoding performance of polar codes (empty marker for SC decoding, and full marker for SC-List decoding), while dashed curves show the decoding performance of LDPC codes. At FER $= 10^{-3}$, we may observe the following:

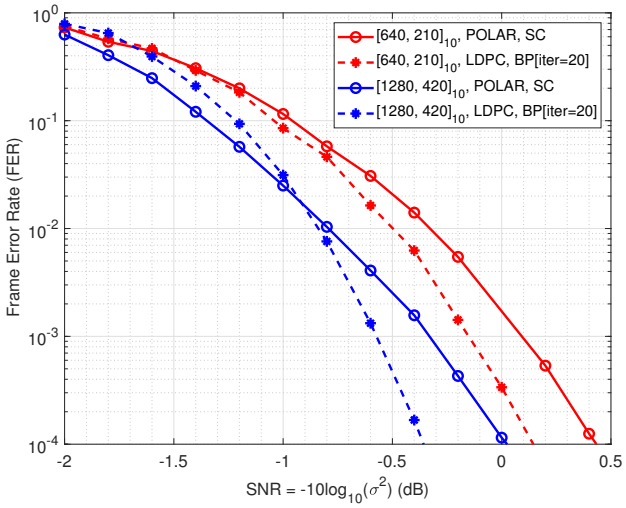
- For codes defined on GF(64), the SC-List decoder outperforms the SC decoder by ≈ 0.6 dB, enclosing in-between the decoding performance of the LDPC code.



(a) Codes defined on GF(64)



(b) Codes defined on GF(256)



(c) Codes defined on GF(1024)

Figure 4.5: FER performance of unpunctured non-binary Polar codes with rate 1/3, over the BI-AWGN channel. Code parameters $[N_{\text{bin}}, K_{\text{bin}}]_p$ are indicated in the legend. The non-binary code-length is given by $N = N_{\text{bin}}/p$, which is a power of 2, thus designed polar codes need not be punctured.

- For codes defined on GF(256), the SC-List decoder outperforms the SC decoder by $\approx 0.5\,\text{dB}$, enclosing in-between, but closer to the SC-List curve, the decoding performance of the LDPC code. The slope of the SC-List decoding curve, especially for the $[1024, 336]_8$ code, indicates a poor minimum distance of the concatenated CRC-Polar code, which may be due to the size of the CRC code (a CRC of size 8 has been used).

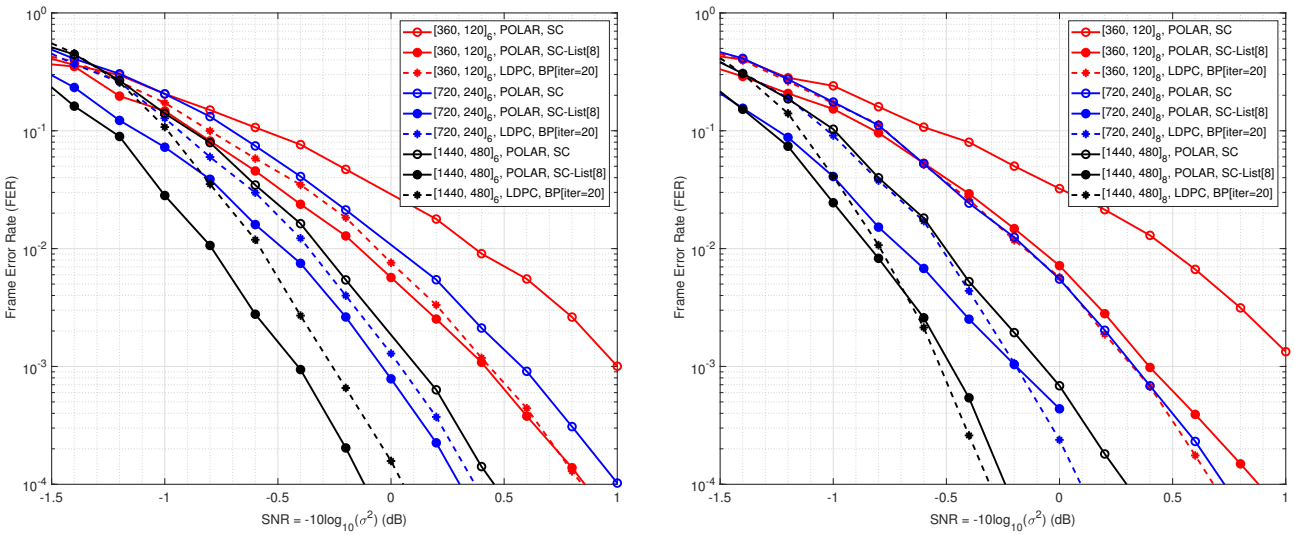
- For codes defined on GF(1024), the SC-List decoder has not been simulated. The LDPC code outperforms the Polar code under SC decoding by $\approx 0.2\,\text{dB}$.

The FER performance of punctured polar codes is shown at Fig. 4.6. We consider codes of rate 1/3, with $K_{\text{bin}} = 120, 240, 480$, defined on either GF(64) or GF(256). At FER $= 10^{-3}$, we may observe the following:

- For codes defined on GF(64), the SC-List decoder outperforms the SC decoder by $\approx 0.5 - 0.6\,\text{dB}$, which is similar to what has been observed in case of unpunctured codes. The performance of the LDPC code is again in between that of SC and SC-List decoders, but the gap with respect to the SC-List decoder is negligible for $K_{\text{bin}} = 120$, and of only 0.1 dB for $K_{\text{bin}} = 240$, and 0.15 dB for $K_{\text{bin}} = 480$.

- For codes defined on GF(256), the SC-List decoder outperforms the SC decoder by $\approx 0.4\,\text{dB}$ for $K_{\text{bin}} = 480$, to more than 0.6 dB for $K_{\text{bin}} = 120$. The slope of the SC-List decoding curve, especially for the $[720, 240]_8$ code, indicates a poor minimum distance of the concatenated CRC-Polar code, which may be due to the size of the CRC code (a CRC of size 8 has been used). The LDPC code slightly outperforms the polar code under SC-List decoding (@FER $= 10^{-3}$).

As a general remark, analyzing the decoding performance of both unpunctured and punctured polar codes, we observe that punctured codes under SC-List decoding exhibit a smaller gain with respect to their counterpart LDPC codes. The gain decrease is most likely a direct consequence of the puncturing process (since LDPC codes are not punctured), indicating that punctured polar codes are less efficient than unpunctured ones.



(a) Codes defined on GF(64)

(b) Codes defined on GF(256)

Figure 4.6: FER performance of punctured non-binary Polar codes with rate 1/3, over the BI-AWGN channel. Code parameters $[N_{\text{bin}}, K_{\text{bin}}]_p$ are indicated in the legend. The non-binary code-length is given by N_{bin}/p, which is not power of 2. Designed polar codes are obtained by puncturing a mother code whose non-binary code-length N is the least power of 2 greater than N_{bin}/p.

4.3.2 AWGN channel with CCSK modulated inputs

Throughout this section, we assume that non-binary coded symbols in GF($q = 2^p$) are mapped into CCSK symbols of length q, which then undergo additive white Gaussian noise[7]. Therefore, we shall distinguish between the *native coding rate* of the non-binary code, denoted by R, and by effective coding rate, defined as $R_{eff} = \frac{p}{q}R$ (see also deliverable D1.1).

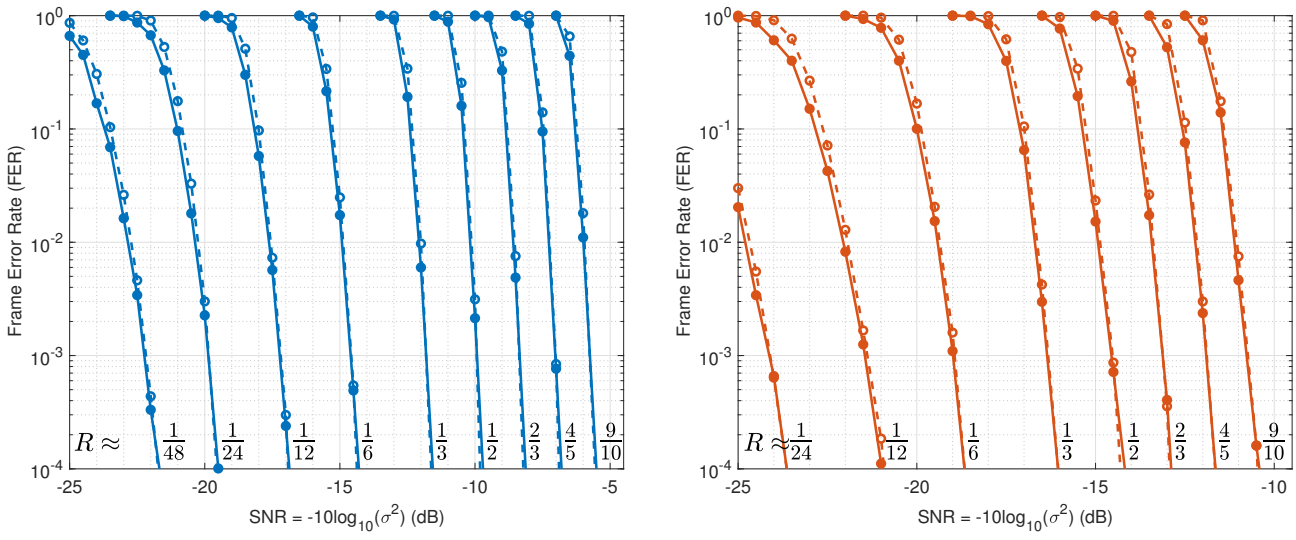
We have considered SNR values from -25 to -5 dB, with a step of 0.5 dB, and for each SNR value we have constructed non-binary polar codes with parameters given in Table 4.1. Recall that n is the number of polarization steps, $N = 2^n$ is the non-binary code length (number of coded symbols), Np is the binary code length (number of coded bits), and Nq is the effective number of transmitted bits (after CCSK modulation). It can be observed that polar code parameters have been chosen so that the effective number of transmitted bits $Nq = 65536$. The number of information bits, denoted K_{bin}, depends on the coding rate, and can be obtained by using

$$K_{\text{bin}} = RNp = R_{eff}Nq \tag{4.14}$$

Table 4.1: Parameters of the non-binary polar codes designed for the for the AWGN channel with CCSK modulated inputs

p	n	N	Np	Nq
6	10	1024	6144	65536
8	8	256	2048	65536
10	6	64	640	65536

Fig. 4.7 shows the FER performance for various native coding rate values R, varying from 1/48 to 9/10. Two FER curves are shown for each native coding rate, a solid one, corresponding to Monte Carlo simulation results, and a dashed one, corresponding to the FER estimated at the code construction stage (Section 4.2.3). It can be observed that the FER estimates we obtain at the code construction stage are tight.

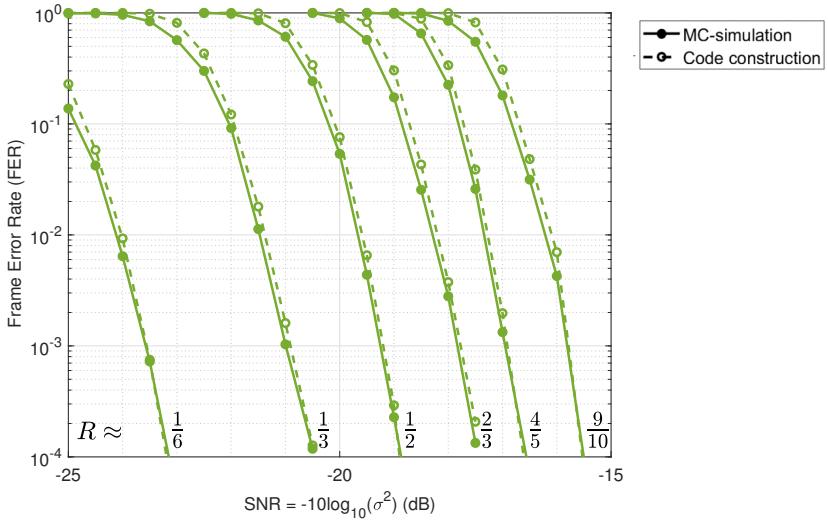


(a) Polar codes defined on GF(64)

(b) Polar codes defined on GF(256)

Figure 4.7: FER performance of non-binary Polar codes over the AWGN channel with CCSK modulated inputs (continued on next page)

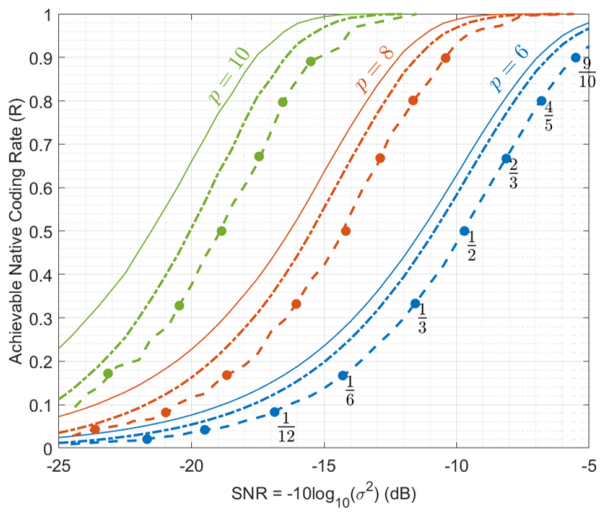
[7]Recall that CCSK symbols belong to $\{\pm 1\}^q$, and we consider real noise, with noise variance σ^2.



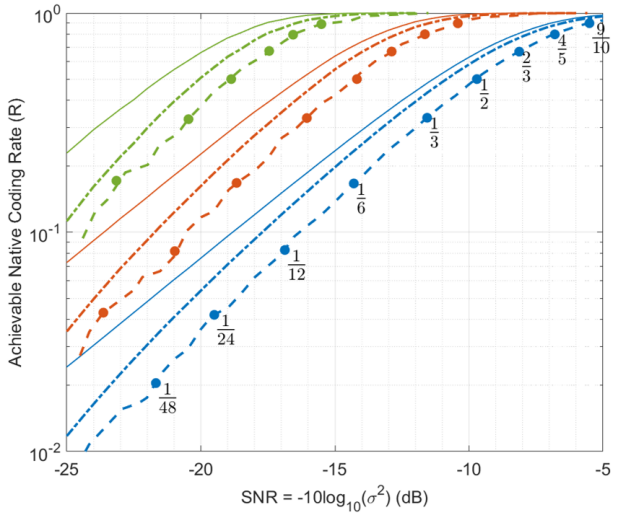
(a) Polar codes defined on GF(1024)

Figure 4.7: FER performance of non-binary Polar codes over the AWGN channel with CCSK modulated inputs, for various native coding rate values R. Solid curves correspond to Monte Carlo simulation results, while dashed curves show the FER estimated at the code construction stage.

Fig. 4.9 shows the achievable native and effective coding rates, for a target FER $= 10^{-4}$. The figure shows the achievable coding rates for different Galois fields, obtained by using either the FER estimates at the code construction stage (dashed curves), or the FER obtained by Monte Carlo simulation (superimposed full markers). Moreover, dashed-dotted curves show the normal approximation of the maximum achievable rate in the finite block-length regime (Polyanskiy-Poor-Verdú bound [38], see also deliverable D1.1), while solid curves show the maximum achievable rate in the asymptotic block-length regime (Shannon limit [39]). It can be seen that the gap between the achievable coding rates under non-binary polar coding and the normal approximation bound is about $1-1.5$ dB. According to the simulation results from the previous section, we expect that this gap can be reduced below 1 dB, by using a SC-List decoder.

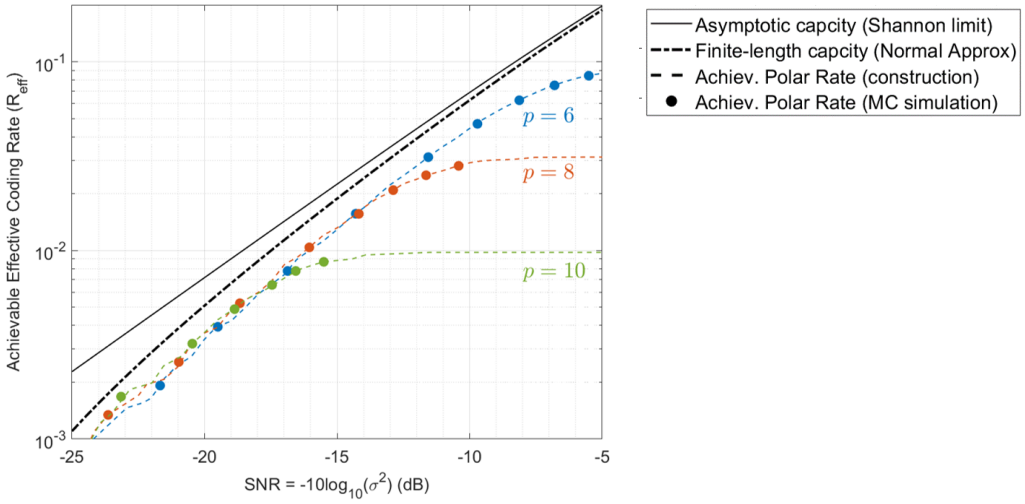


(b) Achievable native coding rates (liner scale)



(c) Achievable native coding rates (log scale)

Figure 4.8: Achievable native and effective coding rates for the AWGN channel with CCSK modulated inputs, for a target FER $= 10^{-4}$ (continued on next page).



(a) Achievable effective coding rates

Figure 4.9: Achievable native and effective coding rates for the AWGN channel with CCSK modulated inputs, for a target FER $= 10^{-4}$.

4.4 Conclusion/summary

This section focused on the design of non-binary polar codes. First, different approaches for polarizing non-binary channels have been discussed (Section 4.1), which motivated a particular choice, consisting in "adjoining" a random GF coefficient to the original kernel introduced by Arikan for polarizing binary-input channels. Then, a design procedure has been proposed (Section 4.2), addressing both the optimization of the non-binary kernel coefficients, and the choice of the virtual channels carrying the information symbols. We have further extended the SC-List decoder to the case of non-binary polar codes, in order to improve the SC decoding performance, especially for short to moderate code-lengths. Different puncturing strategies have also been discussed, supporting the choice of a simple but effective puncturing strategy for non-binary codes. Finally, simulation results have been presented (Section 4.3) for the AWGN channel, with either binary or CCSK modulated inputs. For the BI-AWGN channel, the FER performance of the designed non-binary polar codes under SC decoding is rather modest, noticeably behind that of non-binary LDPC codes (designed for comparison purposes). However, under CRC-assisted SC-List decoding, designed non-binary polar codes provide excellent performance, competing with, and even outperforming, that of non-binary LDPC codes. This result encourages us to pursue our efforts in developing an efficient SC-List decoder for non-binary polar codes. Finally, for the AWGN channel with CCSK modulated inputs, we have shown that the achievable coding rates under SC decoding are within 1 to 1.5 dB from the normal approximation bound, and we expect this gap to be reduced below 1 dB by using a SC-List decoder.

5 General Conclusion

In this deliverable, non-binary forward error correcting codes were designed, intended for association with the CCSK modulation. Three code families were considered for the design: Turbo codes, LDPC codes and polar codes. The design procedure for each of the code families was first described, including the required modifications/additions with respect to the design of classical binary codes of the same family. Rate adaptation procedures were also addressed, including potential support of incremental redundancy.

Then, initial designs for the agreed upon code parameters in D1.1 were provided for an association with binary modulations over an AWGN channel. For some code families and when possible, results for an association with a CCSK modulation were also given.

At a first glance, we can say that the obtained performance by each of the three code families is close. This fact largely motivates the foreseen second design phase with a greater fine-graine tweaking. This second phase is set to be reported in the updated version of this deliverable, namely D1.2b. Complexity and implementation aspects regarding the decoding algorithms and explored in T1.3 are expected to be taken into account for the final selection and recommendations.

Bibliography

[1] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short Turbo Codes over High Order Fields," *IEEE Transactions on Communications*, vol. 61, no. 6, pp. 2201–2211, jun 2013.

[2] T. Matsumine and H. Ochiai, "Capacity-approaching non-binary turbo codes: A hybrid design based on EXIT charts and union bounds," *IEEE Access*, vol. 6, pp. 70 952–70 963, 2018.

[3] R. Klaimi, C. A. Nour, C. Douillard, and J. Farah, "Design of low-complexity convolutional codes over GF(q)," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, dec 2018.

[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[5] R. Garzon-Bohorquez, C. A. Nour, and C. Douillard, "Improving turbo codes for 5g with parity puncture-constrained interleavers," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*. IEEE, sep 2016.

[6] R. G. Bohorquez, C. A. Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 1833–1844, may 2018.

[7] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes: towards a single model," in *2004 IEEE International Conference on Communications.* IEEE, 2004.

[8] R. G. Bohorquez, C. A. Nour, and C. Douillard, "On the equivalence of interleavers for turbo codes," *IEEE Wireless Communications Letters*, vol. 4, no. 1, pp. 58–61, feb 2015.

[9] R. Garzon-Bohorquez, R. Klaimi, C. A. Nour, and C. Douillard, "Mitigating correlation problems in turbo decoders," in *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*. IEEE, dec 2018.

[10] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," *IEEE Communications Letters*, vol. 2, pp. 165–167, 1998.

[11] X. Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph GF(2^q) codes," in *IEEE Int. Conf. on Commun. (ICC)*, Paris, France, June 2004.

[12] L. Dolecek, D. Divsalar, Y. Sun, and B. Amiri, "Non-binary protograph-based ldpc codes: Enumerators, analysis, and designs," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3913–3941, 2014.

[13] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2, d_c)$-ldpc codes over GF(q) using their binary images," *IEEE Trans. on Communications*, vol. 56, no. 10, pp. 1626–1635, 2008.

[14] Ben-Yue Chang, L. Dolecek, and D. Divsalar, "Exit chart analysis and design of non-binary protograph-based ldpc codes," in *2011 - MILCOM 2011 Military Communications Conference*, 2011, pp. 566–571.

[15] D. Divsalar and L. Dolecek, "Graph cover ensembles of non-binary protograph ldpc codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, 2012, pp. 2526–2530.

[16] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788–1793, 2004.

[17] M. Gholami and M. Alinia, "High-performance binary and non-binary low-density parity-check codes based on affine permutation matrices," *IET Communications*, vol. 9, no. 17, pp. 2114–2123, 2015.

[18] E. Boutillon, "Optimization of non binary parity check coefficients," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2092–2100, 2019.

[19] A. Abdmouleh, E. Boutillon, L. Conde-Canencia, C. Abdel Nour, and C. Douillard, "A new approach to optimise non-binary ldpc codes for coded modulations," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 295–299.

[20] V. Savin and D. Declercq, "Linear growing minimum distance of ultra-sparse non-binary cluster-ldpc codes," in *2011 IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 523–527.

[21] D. Declercq, V. Savin, and L. P. Sy, "Analysis and design of ultra-sparse non-binary cluster-ldpc codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, 2012, pp. 2531–2535.

[22] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1365–1375, 2010.

[23] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *IET Electronics Letters*, vol. 46, no. 9, pp. 633–634, 2010.

[24] C. Lin, S. Tu, C. Chen, H. Chang, and C. Lee, "An efficient decoder architecture for nonbinary ldpc codes with extended min-sum algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 9, pp. 863–867, 2016.

[25] U. . G. Fiebig and M. Schnell, "Correlation properties of extended m-sequences," *Electronics Letters*, vol. 29, no. 20, pp. 1753–1755, 1993.

[26] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[27] E. Şaşoğlu, E. Telatar, and E. Arikan, "Polarization for arbitrary discrete memoryless channels," in *IEEE Information Theory Workshop.* IEEE, 2009, pp. 144–148.

[28] A. G. Sahebi and S. S. Pradhan, "Multilevel channel polarization for arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 7839–7857, 2013.

[29] R. Mori and T. Tanaka, "Channel polarization on q-ary discrete memoryless channels by arbitrary kernels," in *IEEE International Symposium on Information Theory.* IEEE, 2010, pp. 894–898.

[30] ——, "Non-binary polar codes using reed-solomon codes and algebraic geometry codes," in *IEEE Information Theory Workshop.* IEEE, 2010, pp. 1–5.

[31] E. Şaşoğlu, "Polar coding theorems for discrete systems," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2011.

[32] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[33] L. Zhang, Z. Zhang, X. Wang, Q. Yu, and Y. Chen, "On the puncturing patterns for punctured polar codes," in *International Symposium on Information Theory.* IEEE, 2014, pp. 121–125.

[34] K. Niu, K. Chen, and J.-R. Lin, "Beyond turbo codes: rate-compatible punctured polar codes," in *International Conference on Communications (ICC).* IEEE, 2013, pp. 3423–3427.

[35] L. Chandesris, V. Savin, and D. Declercq, "On puncturing strategies for polar codes," in *IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 766–771.

[36] L. Chandesris, "Contribution à la construction et au décodage des codes polaires," Ph.D. dissertation, Université de Cergy-Pontoise, 2019.

[37] V. Bioglio, F. Gabry, and I. Land, "Low-complexity puncturing and shortening of polar codes," in *IEEE Wireless Communications and Networking Conference Workshops (WCNC Workshops)*. IEEE, 2017, pp. 1–6.

[38] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[39] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3-4, pp. 379–423 and 623–656, 1948.