Forward Error Correction

Meryem Benammar

10 janvier 2022

Résumé

In the present textbook, we present a quick introduction to forward error correction, with a particular emphasis on linear block codes and their respective decoding algorithms. We touch also briefly on the experimental setups to evaluate the performances of the developped error correction codes and relate the obtained results to Shannon's capacity theorem. A more extensive and rigorous reference on error correction coding can be found in [**?**].

Table des matières

1 Error correction coding : channel coding

In the following, we introduce the notion of error correction coding, also termed *channel coding*, and justify the need behind it in modern communication systems.

1.1 Noisy channels

The notion of noise is paramount to any communication channel between a transmitter and a receiver. Indeed, for all typse of channels, radio frequency channels, optical wired channel, free space optical channels, molecular channel, DNA transcription channels, wired ... there exists some inherent noise, whose level effect could be more or less important, and which impacts the quality of the communication process.

If we were to transmit a message without an error correction mechanism, then, the message would inevitably be plagued with errors.

1.2 Forward error correction Vs feedback error correction

There are two types of error correction mechanisms which can be implemented in a communication scenario, namely, forward error correction and feedback error correction.

- In forward error correction, prior to sending the message, we add to it some form of redundancy which will allow to make sure that, even in the presence of channel errors, the original message can be recovered without errors. In this case, since we add some redundancy to the transmitted messages (repetitions for instance) then we have to sacrifice part of the transmitted rate to convey these redundant bits of messages, and thus, we lose in the useful transmitted rate. On the other side however, error correction allows to have a low probability of error, so a trade-off will arise between the useful rate and the probability of error. Besides, the amount of redundancy needed will depend on the quality of the channel, which we generally estimate prior to transmitting messages.
- In feedback error correction, the message is transmitted with an integrity check mechanism, if the message is received with errors, then the receiver sends a feedback signal to ask for re-trasnmission. Alike forward error correction, in feedback error correction part we degraded the useful packet rate transmitted since we have to retransmit all packets which have been received with errors. However, on the other side, the probability of error improves, and thus, we encounter the same sort of tradeoff between the useful rate and the probability of error. Finally, feedback error correction does not require to estimate the noise level, however, it relies on a dedicated feedback link, which is come applications might not be available.

In this course, we will be interested on in forward error correction mechanisms.

1.3 The channel coding problem

Developping forward error correction mechanisms for a given noisy channel is termed as the channel coding problem, depiected in Figure 1, and it consists in the following problem.

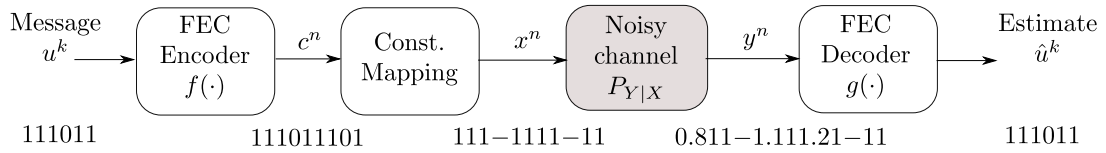

FIGURE 1 – A basic communication chain

Consider a binary message of k bits which we will denote as $u^k = (u_1, ..., u_k)$.

- The encoder $f(\cdot)$ maps the sequence of bits u^k into a coded-message c^n of n bits

$$\begin{array}{rccc} f: & \{0,1\}^k & \rightarrow & \{0,1\}^n \\ & u^k & \rightarrow & c^n = f(u^k) \end{array} \tag{1}$$

- Prior to transitting through a noisy channel, a modulation (constellation mapping) opertation is performed in order to transform the binary coded message c^n into a real-valued sequence of symbols x^n
- The obtained symbols transit then through a noisy channel, which yields a noisy observation y^n (often either with real values, or binary values)
- The decoder $g(\cdot)$ then recovers an estimate $\hat{u}^k$ of the original message given the received sequence y^n

$$\begin{array}{rccc} g: & \mathbb{R}^n & \rightarrow & \{0,1\}^k \\ & y^n & \rightarrow & \hat{u}^k = g(y^n). \end{array} \tag{2}$$

The purpose of error correction coding is to develop an encoder $f(\cdot)$ and a decoder $g(\cdot)$ such that the **probability of error of the messages be arbitrarily small.**

In order to compute this probability of error for all types of messages, we will assume that the message is a string of iid uniform bits (Bern(0.5)), and be interested in two main definitions of error probabilities.

Definition 1 (Block error probability (BLEP)) *The block error probability allows to compute the probability of receiving a wrong packet, i.e., at least one bit wrong in the estimated message, and is defined as*

$$P_{block} = \mathbb{P}(\hat{U}^k \neq U^k) = \mathbb{P}(g(Y^n) \neq U^k) \tag{3}$$

The block error probability is often termed as well, Frame Error Probability (FEP) or Packet Error Probability (PEP). It allows to estimate how many packets are lost in average, however, it does not allow to know exactly how many errors were made per packet in average. To this end, we will also define the following error probability measure.

Definition 2 (Bit Error Probability (BEP)) *Bit error probability allows to compute the probability of receiving a wrong bit, i.e., average number of errors occuring inside a message of length k, and is defined as*

$$P_{bit} = \frac{1}{k}\sum_{i=1}^{k} \mathbb{P}(\hat{U}_i \neq U_i) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{P}(g(Y^n)_i \neq U_i) \tag{4}$$

In practical systems, depending on the nature of the data and of the application, we might be interested in one or the other of error probabilities, sometimes in both error probabilities for rather sensitive data.

In the following, we will differentiate both types of errors when investigating the performances of error correction codes.

1.4 Families of forward error correction codes

In forward error correction (FEC), we can distinguish a few families of error correction codes.

1. Linear block codes. These codes encode information in a block manner using a matrix multiplication as an encoding routine. As such, they are termed linear. Linear block codes have been extensively investigated since Shannon's work in 1948, and constitue to date some of the best error correction codes known. Among these lie the Hamming code, the LDPC codes, the Polar codes, the BCH codes, ...
2. Convolutional codes. These codes encode a stream of bits into a stream of bits using a bank of binary shift registers, and are thus termed convolutional. Convolutional codes are the main block of one of the most powerful codes known which is the Turbo-code.
3. Non-linear codes. Non-linear codes are less widely used than linear codes, because their implementation is quite costy, and are proved optimal for channels which are not very common in practise (molecular channels, optical channels with On Off Keying (OOK), ...)

In this course, we will mainly concentrate on Linear block codes, and investigate three main families of codes.

2 The $GF(2)$ algebra

Error correction is performed at the binary message level, hence the GF(2) algebra. We will define in the following a few useful operations on GF(2).

2.1 Addition and mutliplication

Altough multiplication $(\cdot)$in GF(2) is no different from the one in the set of real numbers, $\mathbb{R}$, we will see that addition $(\oplus)$ is fundamentally different from the classical addition $(+)$. Hereafter are given the two tables of multiplication and addition in GF(2).

$(\cdot)$	0	1
0	0	0
1	0	1

$(\oplus)$	0	1
0	0	1
1	1	0

Since the values $\{0, 1\}$ in GF(2) can be assimiliated to boolean values {True, False}, the multiplication operation can be assimilated to a AND oepration, while the addition opertation can be assimilated to a XOR operation (exclusive OR).

2.2 Vector and matrix products

Let us now consider two vectors of k binary values $u^k = (u_1, ..., u_k)$ and $v^k = (v_1, ..., v_k)$. These vector belongs to the vector space $\mathrm{GF}(2)^k$ on which the addition is defined as follows.

Definition 3 (Sum of two binary vectors) *The binary sum of two vectors $u^k, v^k \in GF(2)^k$ is given by*

$$u^k \oplus v^k = (u_1 \oplus v_1, ..., u_k \oplus v_k). \tag{5}$$

In the following, we will also define a vector product in $GF(2)^k$.

Definition 4 (Vector product) *Let G be a binary matrix of size $(k \times n)$. The vector product $c^n = u^k.G$ of is defined as*

$$c_i = \oplus_{j=1}^{k} u_j.G_{j,i}. \tag{6}$$

2.3 Hamming weight and Hamming distance

In error correction coding, we will often resort to the notions of Hamming weights and Hamming distances. In the following we define both measures.

Definition 5 (Hamming weight and distance) *Let u^k and v^k be two binary vectors. Then*

- *The Hamming distance between u^k and v^k is defined as*

$$d_H(u^k, v^k) = \sum_{i=1}^{k} (u_i \oplus v_i). \tag{7}$$

- *The Hamming weight of a vector u^k is defined as*

$$w_H(u^k) = \sum_{i=1}^{k} u_i. \tag{8}$$

Here the $\sum$ operator is defined with respect to the addition $(+)$ in $\mathbb{R}$, and not in $GF(2)$

Here, a few properties are in order.

Properties 1 *1. d_H is symmetric in u^k and v^k,i.e.,*

$$d_H(u^k, v^k) = d_H(v^k, u^k). \tag{9}$$

2. The hamming distance between two vectors verifies

$$0 \leq d_H(u^k, v^k) \leq k \tag{10}$$

3. The Hamming weight is related to the Hamming distance as follows

$$w_H(u^k) = d_H(u^k, 0^k). \tag{11}$$

Now that we have defined a few tools and operations on binary vectors, we can move on to the definition of a Linear Block code.

3 Linear block codes

Linear block codes are a large family of codes, with long running research and contributions since Shannon's results in 1948. In the following, we span constructions and design criteria for basic linear block codes.

3.1 Encoding and codebook

Linear block codes, are codes which are based on a linear operation in $\mathrm{GF}(2)^k$.

Definition 6 (Linear block codes) *An (n, k) linear block code is a mapping from a binary message of k bits u^k to a binary codeword of length n c^n, through the following linear operation*

$$c^n = u^k.G \tag{12}$$

where G is a $(k \times n)$ binary matrix termed the generator matrix.

This linear operation is performed in the binary field $\mathrm{GF}(2)$, using the multiplication $\times$ and xor operations $\oplus$. Here is a little example :

- Repetition $(3, 1)$ code. The generator matrix for this code is given by

$$G = [111]. \tag{13}$$

  This code maps every single bit u to a triplet of three repetitions of the same bit $c^3 = (u, u, u)$.

- The Hamming $(7, 4)$ code. The generator matric of this code is given by

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{14}$$

  This code maps every message of 4 bits $u^4 = (u_1, u_2, u_3, u_4)$ into a codeword of 7 bits $c^7 = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ given by

$$\begin{aligned} c_1 &= u_1 && (15)\\ c_2 &= u_2 && (16)\\ c_3 &= u_3 && (17)\\ c_4 &= u_4 && (18)\\ c_5 &= u_1 \oplus u_2 \oplus u_4 && (19)\\ c_6 &= u_1 \oplus u_3 \oplus u_4 && (20)\\ c_7 &= u_2 \oplus u_3 \oplus u_4 && (21) \end{aligned}$$

- The $(9, 3)$ Best Know Linear Code (BKLC) whose generator matrix is given by

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{22}$$

  This code maps every message of 3 bits u^3 into a codeword c^9 of 9 bits (equations omitted)

The codebook of a linear block code is the set of the codewords obtained from all possible binary messages. As such, the codebook of an (n, k) linear block code contains 2^k different binary codewords.

Definition 7 (Codebook) *The codebook $\mathcal{C}$ of a linear block code is defined as*

$$\mathcal{C} = \left\{ u^k.G, \text{ for all } u^k \in GF(2)^k \right\}. \tag{23}$$

Often, in litterature, the notion of codebook and the notion of a code are interchangeable, and thus, a code can be defined by its corresponding codebook.

3.2 The parity check matrix

A codebook is a subspace of dimension k of the binary product field $GF(2)^n$ which is of dimension n.

This subspace could also be characterized by its orthogonal space, of dimension $n-k$, called the parity check space. This orthogonal subspace has a generator matrix, called the parity check matrix H of size $(n-k, n)$, and thus,

Definition 8 (Parity check representation) *A linear code can be defined by its parity check matrix as*

$$\mathcal{C} = \left\{ c^n \in GF(2)^n, \text{ such that } H.c^n = 0^{n-k} \right\}. \tag{24}$$

One can show, that this subspace is equal to the subspace defined by the generator matrix, and that is is composed of 2^k binary codewords c^n. Thus, in the error correction community, linear block codes are defined either by their generator matrix G, or their parity check matrix H, which are dual and equivalent.

Here is an example for the $(7, 4)$ Hamming code.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \text{ and } H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{25}$$

One can easily check that any codeword c^7 from the Hamming code, verifies that :

$$c_1 \oplus c_4 \oplus c_5 \oplus c_7 = 0 \tag{26}$$

$$c_2 \oplus c_4 \oplus c_6 \oplus c_7 = 0 \tag{27}$$

$$c_3 \oplus c_5 \oplus c_6 \oplus c_7 = 0. \tag{28}$$

3.3 Decoding : sequence MAP, ML, and bit MAP

Given the received sequence of real-valued channel outputs y^n, the decoder's task is to recover the binary message u^k with as few errors as possible. Let $g(\cdot)$ be the function that associates with each received sequence y^n, an estimate of the message $\hat{u}^k$, given by

$$\hat{u}^k = g(y^n). \tag{29}$$

Note that this function $g(\cdot)$ is perfectly deterministic.

3.4 Sequence MAP and ML decoders

If our target is to minimize the so-called BLock Error Probability (BLEP), defined by

$$BLEP = \mathbb{P}(\hat{U}^k \neq U^k) = \mathbb{P}\left(g\left(\hat{Y}^n\right) \neq U^k\right) \tag{30}$$

then, the optimal decision rule g is the so-called sequence Maximum A Posteriori (MAP) decoder.

Definition 9 (Sequence MAP decoder) *Let y^n be the sequence received at the destination. Then, the sequence MAP decoder is defined by :*

$$\hat{u}^k = g_{MAP}(y^n) = argmax_{u^k} P_{U^k|Y^n}(u^k|y^n). \tag{31}$$

The probability distribution $P_{U^k|Y^n}$ is termed the A Posteriori (AP) distribution.

This means, that the optimal decision rule is the one that associated to each received sequence y^n, the message u^k that is most probable conditioned to it. The probability distribution $P_{U^k|Y^n}$ is termed a A Posteriori (AP) distribution, and is generally unknown to the receiver, and has to be computed which brings about heavy calculations. Indeed, if we write the following AP distribution,

$$P_{U^k|Y^n}(u^k|y^n) = \frac{P_{U^k}(u^k)P_{Y^n|U^k}(y^n|u^k)}{P_{Y^n}(y^n)} \tag{32}$$

then we can clearly see that it involves three terms

- The distribution $P_{U^k}(u^k)$ termed the prior distribution, often taken as uniform 2^{-k}
- The conditional distribution $P_{Y^n|U^k}(y^n|u^k)$ which is often known to the receiver, e.g., a Gaussian distribution
- The marginal distribution $P_{Y^n}(y^n)$ which writes as

$$P_{Y^n}(y^n) = \sum_{v^k} P_{U^k}(v^k)P_{Y^n|U^k}(y^n|v^k), \tag{33}$$

  and involves thus a marginalization over 2^k which entails a huge complexity at the receiver.

Instead, in practise, we can resort to the following simpler decoder.

Definition 10 (Maximum Likelihood decoder) *This decoder is called the Maximum-likelihood (ML) decoder, is defined as*

$$\hat{u}^k = g_{ML}(y^n) = argmax_{u^k} f_{Y^n|X^n}(y^n|x^n(u^k)). \tag{34}$$

and is simpler than the MAP in the sense that it no longer relies on the A Posteriori distribution, but instead, uses only the channel transition pdf $f_{Y^n|X^n}$, which can be known to the receiver.

These two decoders are strongly related, and often, in litterature, authors would not even distinguish both because of the following lemma.

Lemma 1 (Sequence MAP and ML) *If the messages are assumed uniform, i.e., $P_{U^k}(u^k) = 2^{-k}$, then the sequence MAP and ML decoder are equivalent.*

$$argmax_{u^k} P_{U^k|Y^n}(u^k|y^n) = argmax_{u^k} f_{Y^n|X^n}(y^n|x^n(u^k)). \tag{35}$$

For both decoders, the MAP and the ML, the decoder needs to know the modulation used and the FEC used in order to recover the binary message.

Note that, whatever the type of decoder we have seen so far, MAP or ML, we always need an exhaustive search over all possible binary words u^k, which has a complexity in 2^k and becomes unfeasible starting from a few tens of bits. (2^128 is more than the age of the universe).

Hence, in this course, we restrict ourselves to very short blocks, in order for us to be able to implement the sequence MAP decoder. In practical systems, however, the order of magnitudes of (n, k) are far beyond the present lab course, in DVB-S2 for instance, $n = 64800$ and $k = 16200$. So one could wonder how is this possible ? The answer is that, for a few families of linear codes, the code generator matrix presents a certain repetitive pattern, or strong structure, which renders the exhaustive search feasible on smaller spaces than the whole code space. Besides, the existing practical algorithms involve many approximations which allows for a low latency implementation. Hence, practical decoders can be built for longer codes, however, there are only a few codes that allow for this, and this has constituted the main research direction for the error correction coding community.

3.5 Bit MAP decoder

If in turn, our purpose is to maximize the so-called Bit Error Probability (BEP), defined by

$$BEP = \frac{1}{k}\sum_{i=1}^{k} \mathbb{P}(\hat{U}_i \neq U_i) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{P}\left(g\left(\hat{Y}^n\right)_i \neq U_i\right) \tag{36}$$

then the optimal decoder would be the bit-MAP decoder, which is defined as follows.

Definition 11 (Bit-MAP decoder) *The bit-MAP decoder produces k distinct bit estimates $\hat{u}_i$, each given by*

$$\hat{u}_i = g_i(y^n) \quad = \quad argmax_{u_i=0,1} P_{U_i|Y^n}(u_i|y^n). \tag{37}$$

where the probabilitoes $P_{U_i|Y^n}$ are named the bit A-Posteriori distributions.

The bit A-Posteriori distributions $P_{U_i|Y^n}$ are even more challenging to compute than the AP distribution $P_{U^k|Y^n}$, since it involves, for each bit index i, a marginalization over all other possible bit indices in $[1:k]$ other than i, i.e.,

$$P_{U_i|Y^n}(u_i|y^n) = \sum_{u_1} ... \sum_{u_k} P_{U^k|Y^n}(u^k|y^n) \tag{38}$$

Often, the bit-MAP decoder cannot be implemented for generic block codes, unless the code presents some nice structure, alike convolution code. In this case, we can implement a bit-MAP decoder called the BCJR akgorithm, and it is based on a Trellis representation of a convolutional code. In this course, we will not be able to tackle this decoder since its theoretic and practical complexity will not fit in the course volume.

4 Performance assessment

In this section, we answer two main questions : what performance measure should we evaluate for a FEC code ? How do we compute this performance measure ?and versus which parameter of interest should we plot these measures ?

The answers to this question is that we mainly look at the BER and BLER curves versus the ratio E_b/N_0 computed through a Monte Carlo simulation. In the following, we elaborate on these three elements.

4.1 Monte Carlo simulations

Monte Carlo simulations are the basic theoretical and practical tool used in communication engineering, and more generally, in signal processing and electrical engineering.

The main principle of Monte-Carlo simulations is the law of large numbers, and it writes as follows.

Definition 12 *Let us draw N_s vectors of size k each x_i^k randomly generated following a certain law P_{X^k}, where $i \in [1 : N_s]$. And let $f(X^k)$ be a certain function such that we are interested in the mathematical expectation $\mathbb{E}_{X^k} f(X^k)$. Then one can show that the empirical average estimator given by*

$$\bar{E}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} f(x_i^k) \tag{39}$$

converges towards the mathematical expectation as N_s grows infinite, i.e.,

$$\lim_{N_s \to \infty} \bar{E}_{N_s} = \lim_{N_s \to \infty} \frac{1}{N_s} \sum_{i=1}^{N_s} f(x_i^k) = \mathbb{E}_{X^k} f(X^k). \tag{40}$$

This means, that if we come across a mathematical expectation $\mathbb{E}_{X^k} f(X^k)$ which is hard to compute because of the size of all possible x^k, then, one can approximate this exepectation with an empirical average estimator provided that the data x^k are generated following the law P_{X^k}. This is the so-called Monte Carlo simulation.

4.2 Error probability estimators : BLER and BER

It is common in FEC to evaluate two measures : the Bit Error Probability (BEP) and the Block Error Probability (BLEP). Let us now write rigorously what these probabilities represent, and take for instance the BLEP :

$$\begin{aligned} BLEP = \mathbb{P}(\hat{U}^k \neq U^k) &= \mathbb{P}\left(g\left(\hat{Y}^n\right) \neq U^k\right) && (41)\\ &= \mathbb{E}_{Y^n, U^k} \mathbb{1}\left(g\left(\hat{Y}^n\right) \neq U^k\right) && (42)\\ &= \mathbb{E}_{U^k} \mathbb{E}_{Y^n|U^k} \mathbb{1}\left(g\left(\hat{Y}^n\right) \neq U^k\right) && (43) \end{aligned}$$

where we have used the fact that a probability of an event, is the expectation of the indicator of this event, and where we have written the expectation over all possible sources of randomness, namely, the message U^k and the channel output Y^n (due to the random noise). However, given the sizes of the codewords and of the binary messages, one cannot

evaluate theoretically this expectation $\mathbb{E}_{Y^n,U^k}$ and hence can only implement an estimator for it. This estimator is the BLock Error Rate estimator (BLER) and is obtained through a so-called Monte-Carlo simulation wherein the expectation $\mathbb{E}_{U^k}\mathbb{E}_{Y^n|U^k}$ of this indicator, is computed as an empirical average over N_s pairs (U^k, Y^n) generated as follows : U^k generated as a uniform binary variable of length k and Y^n generated as the result of passing the sequence $x^n(U^k)$ through the additive noise channel. A block error is counted whenever at least one bit is wrong between U^k and $\hat{U}^k$.

As for the Bit Error Probability, one can write exacly the same expressions,

$$BEP = \frac{1}{k}\sum_{i=1}^{k}\mathbb{P}(\hat{U}_i \neq U_i) = \frac{1}{k}\sum_{i=1}^{k}\mathbb{P}\left(g\left(\hat{Y}^n\right)_i \neq U_i\right) \quad (44)$$

$$= \mathbb{E}_{Y^n,U^k}\frac{1}{k}\sum_{i=1}^{k}\mathbb{1}\left(g\left(\hat{Y}^n\right)_i \neq U_i\right) \quad (45)$$

$$= \mathbb{E}_{U^k}\mathbb{E}_{Y^n|U^k}\frac{1}{k}\sum_{i=1}^{k}\mathbb{1}\left(g\left(\hat{Y}^n\right)_i \neq U_i\right) \quad (46)$$

and hence, the Bit Error Estimator (BER) consists in computing the expectation through a Monte Carlo simulation, by taking counting the number of all bit errors $\hat{U}_i \neq U_i$ in the received sequence.

4.3 Signal to noise ratio E_b/N_0

Now that we have defined the measure to be evaluated and the computation method, let us define the parameter of interest which is the E_b/N_0.

To this end, we will have first to define the so called E_s/N_0 ratio.

$$E_s/N_0 = \frac{\text{Energy consumed per transmitted symbol}}{\text{Noise power spectral density}}. \quad (47)$$

which is paramount to a sort of signal to noise ration. Indeed, to simplify things, we assume that the noise energy is given by $N_0 = 2\sigma^2$, where σ^2 is the variance of the noise generated, and for a BPSK modulation, $E_s = 1$, hence,

$$E_s/N_0 = \frac{1}{2\sigma^2}. \quad (48)$$

The ratio E_s/N_0does not depend on the code parameter, but rather on the modulation used, and the noise variances. As such, the BER and BLER curves are not plot against the E_s/N_0 but rather against the ratio E_b/N_0 which can be defined as

$$E_b/N_0 = \frac{\text{Energy consumed per useful bit}}{\text{Noise power spectral density}} = \frac{n}{k}E_s/N_0. \quad (49)$$

The fact that the E_b/N_0 ratio accounts for the code rate allows for a fair comparison between many coding schemes, by fixing the total amount of power used per transmitted bit at the same level.

Hence, we will mainly look at the BER and BLER curves Vs E_b/N_0 in this lab course. Of course, these curves will fail to compare the codes from a fair transmitted bit-rate point of view.