

# Density Evolution and Functional Threshold for the Noisy Min-Sum Decoder

(Extended Version)

Christiane L. Kameni Ngassa\*<sup>#</sup>, Valentin Savin\*, Elsa Dupraz<sup>#</sup>, David Declercq<sup>#</sup>

\*CEA-LETI, Minatec Campus, Grenoble, France

{christiane.kameningassa, valentin.savin}@cea.fr

<sup>#</sup>ETIS, ENSEA / CNRS UMR-8051 / Univ. Cergy-Pontoise, France

{elsa.dupraz, declercq}@ensea.fr

## Abstract

This paper investigates the behavior of the Min-Sum decoder running on noisy devices. The aim is to evaluate the robustness of the decoder in the presence of computation noise, *e.g.* due to faulty logic in the processing units, which represents a new source of errors that may occur during the decoding process. To this end, we first introduce probabilistic models for the arithmetic and logic units of the finite-precision Min-Sum decoder, and then carry out the density evolution analysis of the noisy Min-Sum decoder. We show that in some particular cases, the noise introduced by the device can help the Min-Sum decoder to escape from fixed points attractors, and may actually result in an increased correction capacity with respect to the noiseless decoder. We also reveal the existence of a specific threshold phenomenon, referred to as functional threshold. The behavior of the noisy decoder is demonstrated in the asymptotic limit of the code-length – by using “noisy” density evolution equations – and it is also verified in the finite-length case by Monte-Carlo simulation.

## I. INTRODUCTION

In traditional models of communication or storage systems with error correction coding, it is assumed that the operations of an error correction encoder and decoder are deterministic and that the randomness

This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (*i*-RISC project).

This paper is an extended version of the paper with same title, submitted to IEEE Transactions on Communications

exists only in the transmission or storage channel. However, with the advent of nanoelectronics, the reliability of the forthcoming circuits and computation devices is becoming questionable. It is then becoming crucial to design and analyze error correcting decoders able to provide reliable error correction even if they are made of unreliable components.

Except the pioneered works by Taylor and Kuznetsov on reliable memories [1]–[3], later generalized in [4], [5] to the case of hard-decision decoders, this new paradigm of noisy decoders has merely not been addressed until recently in the coding literature. However, over the last years, the study of error correcting decoders, especially Low-Density Parity-Check (LDPC) decoders, running on noisy hardware attracted more and more interest in the coding community. In [6] and [7] hardware redundancy is used to develop fault-compensation techniques, able to protect the decoder against the errors induced by the noisy components of the circuit. In [8], a class of modified Turbo and LDPC decoders has been proposed, able to deal with the noise induced by the failures of a low-power buffering memory that stores the input soft bits of the decoder. Very recently, the characterization of the effect of noisy processing on message-passing iterative LDPC decoders has been proposed. In [9], the concentration and convergence properties were proved for the asymptotic performance of noisy message-passing decoders, and density evolution equations were derived for the noisy Gallager-A and Belief-Propagation decoders. In [10]–[12], the authors investigated the asymptotic behavior of the noisy Gallager-B decoder defined over binary and non-binary alphabets. The Min-Sum decoding under unreliable message storage has been investigated in [13], [14]. However, all these papers deal with very simple error models, which emulate the noisy implementation of the decoder, by passing each of the exchanged messages through a noisy channel.

In this work we focus on the Min-Sum decoder, which is widely implemented in real communication systems. In order to emulate the noisy implementation of the decoder, probabilistic error models are proposed for its arithmetic components (adders and comparators). The proposed probabilistic components are used to build the noisy finite-precision decoders. We further analyze the asymptotic performance of the noisy Min-Sum decoder, and provide useful regions and target-BER-thresholds [9] for a wide range of parameters of the proposed error models. We also highlight a wide variety of more or less conventional behaviors and reveal the existence of a specific threshold phenomenon, which is referred to as *functional threshold*. Finally, the asymptotic results are also corroborated through finite length simulations.

The remainder of the paper is organized as follows. Section II gives a brief introduction to LDPC codes and iterative decoding. Section III presents the error models for the arithmetic components. The density evolution equations and asymptotic analysis of the noisy finite-precision Min-Sum decoding are presented in Section V and Section VI respectively. Section VII provides the finite-length performance

and Section VIII concludes the paper.

## II. LDPC CODES AND THE MIN-SUM ALGORITHM

### A. LDPC Codes

LDPC codes [15] are linear block codes defined by sparse parity-check matrices. They can be advantageously represented by bipartite (Tanner) graphs [16] and decoded by message-passing (MP) iterative algorithms. The Tanner graph of an LDPC code is a bipartite graph  $\mathcal{H}$ , whose adjacency matrix is the parity-check matrix  $H$  of the code. Accordingly,  $\mathcal{H}$  contains two types of nodes:

- *variable-nodes*, corresponding to the columns of  $H$ , or equivalently to the codeword bits, and
- *check-nodes*, corresponding to the rows of  $H$ , or equivalently to the parity equations the codeword bits are checked by.

We consider an LDPC code defined by a Tanner graph  $\mathcal{H}$ , with  $N$  variable-nodes and  $M$  check-nodes. Variable-nodes are denoted by  $n \in \{1, 2, \dots, N\}$ , and check-nodes by  $m \in \{1, 2, \dots, M\}$ . We denote by  $\mathcal{H}(n)$  and  $\mathcal{H}(m)$  the *set of neighbor nodes* of the variable-node  $n$  and of the check-node  $m$ , respectively. The number of elements of  $\mathcal{H}(n)$  (or  $\mathcal{H}(m)$ ) is referred to as the *node-degree*.

The Tanner graph representation allows reformulating the *probabilistic decoding* initially proposed by Gallager [15] in terms of Belief-Propagation<sup>1</sup> (BP): an MP algorithm proposed by J. Pearl in 1982 [17] to perform Bayesian inference on trees, but also successfully used on general graphical models [18]. The BP decoding is known to be optimal on cycle-free graphs (in the sense that it outputs the maximum a posteriori estimates of the coded bits), but can also be successfully applied to decode linear codes defined by graphs with cycles, which is actually the case of all practical codes. However, in practical applications, the BP algorithm might be disadvantaged by its computational complexity and its sensitivity to the channel noise density estimation (inaccurate estimation of the channel noisy density may cause significant degradation of the BP performance).

### B. Min-Sum Decoding

One way to deal with complexity and numerical instability issues is to simplify the computation of messages exchanged within the BP decoding. The most complex step of the BP decoding is the computation of check-to-variable messages, which makes use of computationally intensive hyperbolic tangent functions. The Min-Sum (MS) algorithm is aimed at reducing the computational complexity of

<sup>1</sup>Also referred to as Sum-Product (SP)

the BP, by using max-log approximations of the parity-check to coded-bit messages [19]–[21]. The only operations required by the MS decoding are additions, comparisons, and sign ( $\pm 1$ ) products, which solves the complexity and numerical instability problems. The performance of the MS decoding is also known to be independent of the channel noise density estimation, for most of the usual channel models.

For the sake of simplicity, we only consider transmissions over *binary-input* memoryless noisy channels, and assume that the channel input alphabet is  $\{-1, +1\}$ , with the usual convention that  $+1$  corresponds to the 0-bit, and  $-1$  corresponds to the 1-bit. We further consider a codeword  $\underline{x} = (x_1, \dots, x_N) \in \{-1, +1\}^N$  and denote by  $\underline{y} = (y_1, \dots, y_N)$  the received word. The following notation will be used throughout the paper, with respect to message passing decoders:

- $\gamma_n$  is the log-likelihood ratio (LLR) value of  $x_n$  according to the received  $y_n$  value; it is also referred to as the *a priori information* of the decoder concerning the variable-node  $n$ ;
- $\tilde{\gamma}_n$  is the *a posteriori information* (LLR value) of the decoder concerning the variable-node  $n$ ;
- $\alpha_{m,n}$  is the variable-to-check message sent from variable-node  $n$  to check-node  $m$ ;
- $\beta_{m,n}$  is the check-to-variable message sent from check-node  $m$  to variable-node  $n$ .

The (infinite precision) MS decoding is described in Algorithm 1. It consists of an initialization step (in which variable-to-check messages are initialized according to the a priori information of the decoder), followed by an iteration loop, where each iteration comprises three main steps as follows:

- **CN-processing** (check-node processing step): computes the check-to-variable messages  $\beta_{m,n}$ ;
- **VN-processing** (variable-node processing step): computes the variable-to-check messages  $\alpha_{m,n}$ ;
- **AP-update** (a posteriori information update step): computes the a posteriori information  $\tilde{\gamma}_n$ .

Moreover, each iteration also comprises a *hard decision* step, in which each transmitted bit is estimated according to the sign of the a posteriori information, and a *syndrome check* step, in which the syndrome of the estimated word is computed. The MS decoding stops when whether the syndrome is  $+1$  (the estimated word is a codeword) or a maximum number of iterations is reached.

The a priori information (LLR) of the decoder is defined by  $\gamma_n = \log \frac{\Pr(x_n = +1 | y_n)}{\Pr(x_n = -1 | y_n)}$ , and for the two following channel models (predominantly used in this work), it can be computed as follows:

- For the Binary Symmetric Channel (BSC),  $\underline{y} \in \{-1, +1\}^N$  is obtained by flipping each entry of  $\underline{x}$  with some probability  $\varepsilon$ , referred to as the channel's crossover probability. Consequently:

$$\gamma_n = \log \left( \frac{1 - \varepsilon}{\varepsilon} \right) y_n \quad (1)$$

- For the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel,  $\underline{y} \in \mathbb{R}^N$  is obtained by

---

**Algorithm 1** Min-Sum (MS) decoding
 

---

 Input:  $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$  ( $\mathcal{Y}$  is the channel output alphabet) ▷ received word

 Output:  $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{-1, +1\}^N$  ▷ estimated codeword
**Initialization**
**for all**  $n = 1, \dots, N$  **do**  $\gamma_n = \log \frac{\Pr(x_n = +1 | y_n)}{\Pr(x_n = -1 | y_n)}$ ;

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do**  $\alpha_{m,n} = \gamma_n$ ;
**Iteration Loop**
**for all**  $m = 1, \dots, M$  and  $n \in \mathcal{H}(m)$  **do** ▷ **CN-processing**

$$\beta_{m,n} = \left( \prod_{n' \in \mathcal{H}(m) \setminus n} \text{sgn}(\alpha_{m,n'}) \right) \left( \min_{n' \in \mathcal{H}(m) \setminus n} |\alpha_{m,n'}| \right);$$

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do** ▷ **VN-processing**

$$\alpha_{m,n} = \gamma_n + \sum_{m' \in \mathcal{H}(n) \setminus m} \beta_{m',n};$$

**for all**  $n = 1, \dots, N$  **do** ▷ **AP-update**

$$\tilde{\gamma}_n = \gamma_n + \sum_{m \in \mathcal{H}(n)} \beta_{m,n};$$

**for all**  $\{v_n\}_{n=1, \dots, N}$  **do**  $\hat{x}_n = \text{sgn}(\tilde{\gamma}_n)$ ; ▷ hard decision
**if**  $\hat{\underline{x}}$  is a codeword **then** exit the iteration loop ▷ syndrome check
**End Iteration Loop**


---

 $y_n = x_n + z_n$ , where  $z_n$  is the white Gaussian noise with variance  $\sigma^2$ . It follows that:

$$\gamma_n = \frac{2}{\sigma^2} y_n \tag{2}$$

**Remark:** It can be easily seen that if the a priori information vector  $\underline{\gamma} = (\gamma_1, \dots, \gamma_N)$  is multiplied by a constant value, this value will factor out from all the processing steps in Algorithm 1 (throughout the decoding iterations), and therefore it will not affect in any way the decoding process. It follows that for both the BSC and BI-AWGN channel models, one can simply define the a priori information of the decoder by  $\gamma_n = y_n, \forall n = 1, \dots, N$ .

### III. ERROR INJECTION AND PROBABILISTIC MODELS FOR NOISY COMPUTING

#### A. Noisy Message-Passing decoders

The model for noisy MP decoders proposed in [9] incorporates two different sources of noise: *computation noise* due to noisy logic in the processing units, and *message-passing noise* due to noisy wires

(or noisy memories) used to exchange messages between neighbor nodes.

- The computation noise is modeled as a random variable, which the variable-node or the check-node processing depends on. Put differently, an outgoing message from a (variable or check) node depends not only on the incoming messages to that node (including the a priori information for the variable-node processing), but also on the realization of a random variable which is assumed to be independent of the incoming messages.
- The message-passing noise is simply modeled as a noisy channel. Hence, transmitting a message over a noisy wire is emulated by passing that message through the corresponding noisy channel.

However, in [9] it has been noted that “*there is no essential loss of generality by combining computation noise and message-passing noise into a single form of noise*” (see also [22, Lemma 3.1]). Consequently, the approach adopted has been to merge noisy computation into message-passing noise, and to emulate noisy decoders by passing the exchanged messages through different noisy channel models. Thus, the noisy Gallager-A decoder has been emulated by passing the exchanged messages over independent and identical BSC wires, while the noisy BP decoder has been emulated by corrupting the exchanged messages with bounded and symmetrically distributed additive noise (*e.g.* uniform noise or truncated Gaussian noise).

The approach we follow in this work differs from the one in [9] in that the computation noise is modeled at the lower level of *arithmetic and logic operations* that compose the variable-node and check-node processing units. This finer-grained noise modeling is aimed at determining the level of noise that can be tolerated in each type of operation. As the main focus of this work is on computation noise, we shall consider that messages are exchanged between neighbor nodes through error-free wires (or memories). However, we note that *this work can readily be extended to include different error models for the message-passing noise* (as defined in [9]). Alternatively, we may assume that the message-passing noise is merged into the computation noise, in the sense that adding noise in wires would modify the probabilistic model of the noisy logic or arithmetic operations.

### B. Error Injection Models

We only consider the case of finite-precision operations, meaning that the inputs (operands) and the output of the operator are assumed to be bounded integer numbers. We simulate a noisy operator by injecting errors into the output of the noiseless one. In the following,  $\mathcal{V} \subset \mathbb{Z}$  denotes a finite set consisting of all the possible outputs of the noiseless operator.

*Definition 1:* An error injection model on  $\mathcal{V}$ , denoted by  $(\mathcal{E}, p_{\mathcal{E}}, \iota \mid \mathcal{V})$ , is given by:

- A finite error set  $\mathcal{E} \subset \mathbb{Z}$  together with a probability mass function  $p_{\mathcal{E}} : \mathcal{E} \rightarrow [0, 1]$ , referred to as the error distribution;
- A function  $\iota : \mathcal{V} \times \mathcal{E} \rightarrow \mathcal{V}$ , referred to as the error injection function.

For a given set of inputs, the output of the noisy operator is the random variable defined by  $\iota(v, e)$ , where  $v \in \mathcal{V}$  is the corresponding output of the noiseless operator, and  $e$  is drawn randomly from  $\mathcal{E}$  according to the probability distribution  $p_{\mathcal{E}}$ .

The error injection probability is defined by

$$p_0 = \frac{1}{|\mathcal{V}|} \sum_v \sum_e \bar{\delta}_{\iota(v,e)}^v p_{\mathcal{E}}(e), \quad (3)$$

where  $\bar{\delta}_{\iota(v,e)}^v = 0$  if  $v = \iota(v, e)$ , and  $\bar{\delta}_{\iota(v,e)}^v = 1$  if  $v \neq \iota(v, e)$ . In other words,  $p_0 = \Pr(v \neq \iota(v, e))$ , assuming that  $v$  is drawn uniformly from  $\mathcal{V}$  and  $e$  is drawn from  $\mathcal{E}$  according to  $p_{\mathcal{E}}$ .

The above definition makes some implicit assumptions which are discussed below.

- The set of possible outputs of the noisy operator is the same as the set of possible outputs of the noiseless operator ( $\mathcal{V}$ ). This is justified by the fact that, in most common cases,  $\mathcal{V}$  is the set of all (signed or unsigned) integers that can be represented by a given number of bits. Thus, error injection will usually alter the bit values, but not the number of bits.
- The injected error does not depend on the output value of the noiseless operator and, consequently, neither on the given set of inputs. In other words, the injected error is independent on the data processed by the noiseless operator. The validity of this assumption does actually depend on the size of the circuit implementing the operator. Indeed, this assumption tends to hold fairly well for large circuits [23], but becomes more tenuous as the circuit size decreases.

Obviously, it would be possible to define more general error injection models, in which the injected error would depend on the data (currently and/or previously) processed by the operator. Such an error injection model would certainly be more realistic, but it would also make it very difficult to analytically characterize the behavior of noisy MP decoders. As a side effect, the decoding error probability would be dependent on the transmitted codeword, which would prevent the use of the *density evolution* technique for the analysis of the asymptotic decoding performance (since the density evolution technique relies on the all-zero codeword assumption).

However, the fact that the error injection model is data independent does not guarantee that the decoding error probability is independent of the transmitted codeword. In order for this to happen, the error injection model must also satisfy a *symmetry condition* that can be stated as follows.

*Definition 2:* An error injection model  $(\mathcal{E}, p_{\mathcal{E}}, \iota | \mathcal{V})$  is said to be *symmetric* if  $\mathcal{V}$  is symmetric around the origin (meaning that  $v \in \mathcal{V} \Leftrightarrow -v \in \mathcal{V}$ , but 0 does not necessarily belong to  $\mathcal{V}$ ), and the following equality holds

$$\sum_{\{e | \iota(v,e)=w\}} p_{\mathcal{E}}(e) = \sum_{\{e | \iota(-v,e)=-w\}} p_{\mathcal{E}}(e), \quad \forall v, w \in \mathcal{V} \quad (4)$$

The meaning of the symmetry condition is as follows. Let  $V$  be a random variable on  $\mathcal{V}$ . Let  $\phi_V^{(\iota)}$  and  $\phi_{-V}^{(\iota)}$  denote the probability mass functions of the random variables obtained by injecting errors in the output of  $V$  and  $-V$ , respectively. Then the above symmetry condition is satisfied if and only if for any  $V$  the following equality holds

$$\phi_V^{(\iota)}(w) = \phi_{-V}^{(\iota)}(-w), \quad \forall w \in \mathcal{V} \quad (5)$$

A particular case in which the symmetry condition is fulfilled is when  $\iota(-v, e) = -\iota(v, e)$ , for all  $v \in \mathcal{V}$  and  $e \in \mathcal{E}$ . In this case, the error injection model is said to be *highly symmetric*.

Messages exchanged within message-passing decoders are generally in *belief-format*, meaning that the sign of the message indicates the bit estimate and the magnitude of the message the confidence level. As a consequence, errors occurring on the sign of the exchanged messages are expected to be more harmful than those occurring on their magnitude. This motivates the following definition, which will be used in the following section (see also the discussion in Section IV-C).

*Definition 3:* An error injection model  $(\mathcal{E}, p_{\mathcal{E}}, \iota | \mathcal{V})$  is said to be *sign-preserving* if for any  $v \in \mathcal{V}$  and  $e \in \mathcal{E}$ ,  $v$  and  $\iota(v, e)$  are either both non-negative ( $\geq 0$ ) or both non-positive ( $\leq 0$ ).

### C. Bitwise-XOR Error Injection

We focus now on the two main symmetric error injection models that will be used in this work. Both models are based on a bitwise XOR operation between the noiseless output  $v$  and the error  $e$ . The two models differ in the definition of the error set  $\mathcal{E}$ , which is chosen such that the bitwise XOR operation may or may not affect the sign of the noiseless output. In the first case, the bitwise XOR error injection model is said to be *full-depth*, while in the second it is said to be *sign-preserving*. These error injection models are rigorously defined below.

In the following, we fix  $\theta \geq 2$  and set  $\mathcal{V} = \{-\Theta, \dots, -1, 0, +1, \dots, +\Theta\}$ , where  $\Theta = 2^{\theta-1} - 1 \geq 1$ . We also fix a *signed number binary representation*, which can be any of the *sign-magnitude*, *one's complement*, or *two's complement* representation. There are exactly  $2^\theta$  signed numbers that can be



represented by  $\theta$  bits in any of the above formats, one of which does not belong to  $\mathcal{V}$  (note that  $\mathcal{V}$  contains only  $2\Theta + 1 = 2^\theta - 1$  elements for symmetry reasons!). We denote this element by  $\zeta$ . Hence:

- In sign-magnitude format,  $\zeta = -0$ , with binary representation  $10 \cdots 0$ ;
- In one's complement format,  $\zeta = -0$ , with binary representation  $11 \cdots 1$ ;
- In two's complement format,  $\zeta = -(\Theta + 1)$ , with binary representation  $10 \cdots 0$ .

For any  $u, v \in \mathcal{V}$ , we denote by  $u \wedge v$  the bitwise XOR operation between  $u$  and  $v$ . From the above discussion, it follows that  $u \wedge v \in \mathcal{V} \cup \{\zeta\}$ .

1) *Full-depth error injection*: For this error model, the error set is  $\mathcal{E} = \mathcal{V}$ . The error injection probability is denoted by  $p_0$ , and all the possible error values  $e \neq 0$  are assumed to occur with the same probability (for symmetry reasons). It follows that the error distribution function is given by  $p_{\mathcal{E}}(0) = 1 - p_0$  and  $p_{\mathcal{E}}(e) = \frac{p_0}{2^\Theta}$ ,  $\forall e \neq 0$ . Finally, the error injection function is defined by:

$$\iota(v, e) = \begin{cases} v \wedge e, & \text{if } v \wedge e \in \mathcal{V} \\ e, & \text{if } v \wedge e = \zeta \end{cases} \quad (6)$$

2) *Sign-preserving error injection*: For this error model, the error set is  $\mathcal{E} = \{0, +1, \dots, +\Theta\}$ . The error injection probability is denoted by  $p_0$ , and all the possible error values  $e \neq 0$  are assumed to occur with the same probability (for symmetry reasons). It follows that the error distribution function is given by  $p_{\mathcal{E}}(0) = 1 - p_0$  and  $p_{\mathcal{E}}(e) = \frac{p_0}{\Theta}$ ,  $\forall e \neq 0$ . Finally, the error injection function is defined by:

$$\iota(v, e) = \begin{cases} v \wedge e, & \text{if } v \neq 0 \text{ and } v \wedge e \in \mathcal{V} \\ \pm e, & \text{if } v = 0 \\ 0, & \text{if } v \wedge e = \zeta \end{cases} \quad (7)$$

In the above definition,  $\iota(0, e)$  is randomly set to either  $-e$  or  $+e$ , with equal probability (this is due once again to symmetry reasons). Note also that the last two conditions, namely  $v = 0$  and  $v \wedge e = \zeta$ , cannot hold simultaneously (since  $e \neq \zeta$ ).

Finally, we note that both of the above models are highly symmetric, if one of the sign-magnitude or the one's complement representation is used. In case that the two's complement representation is used, they are both symmetric, but not highly symmetric.

An example of sign-preserving bitwise-XOR error injection is given in Table I. The number of bits is  $\theta = 5$  and two's complement binary representation is used. The sign bit of the error is not displayed, as it is equal to zero for any  $e \in \mathcal{E}$ . The positions of 1's in the binary representation of  $e$  correspond to the positions of the erroneous bits in the noisy output.

Table I  
EXAMPLE OF SIGN-PRESERVING BITWISE-XOR ERROR INJECTION

	integer	2's complement binary representation				
noiseless output: $v$	-11	1	0	1	0	1
error: $e$	6		0	1	1	0
noisy output: $\iota(v, e)$	-13	1	0	0	1	1
bit position		$\theta=5$	4	3	2	1

**Remark:** It is also possible to define a *variable depth error injection* model, in which errors are injected in only the  $\lambda$  least significant bits, with  $\lambda \leq \theta$ . Hence,  $\lambda = \theta$  corresponds to the above full-depth model, while  $\lambda = \theta - 1$  corresponds to the sign-preserving model. However, for  $\lambda < \theta - 1$  such a model is **not** symmetric, if the two's complement representation is used.

#### D. Output-Switching Error Injection

A particular case is represented by error injection on binary output. Assuming that  $\mathcal{V} = \{0, 1\}$ , the *bit-flipping* error injection model is defined as follows. The error set is  $\mathcal{E} = \{0, 1\}$ , with error distribution function given by  $p_{\mathcal{E}}(0) = 1 - p_0$  and  $p_{\mathcal{E}}(1) = p_0$ , where  $p_0$  is the error injection probability, and the error injection function is given by  $\iota(v, e) = v \wedge e$ . Put differently, the error injection model flips the value of a bit in  $\mathcal{V}$  with probability  $p_0$ .

Clearly, the above error injection model can be applied on any set  $\mathcal{V}$  with two elements, by switching one value to another with probability  $p_0$ . In this case, we shall refer to this error injection model as *output-switching*, rather than bit-flipping.

Moreover, if one takes  $\mathcal{V} = \{-1, +1\}$  (with the usual 0,1 to  $\pm 1$  conversion), it can be easily verified that this error injection model is highly symmetric.

#### E. Probabilistic models for noisy adders, comparators and XOR-gates

In this section we describe the probabilistic models for noisy adders, comparators and xor-gates, built upon the above error injection models. These probabilistic models will be used in the next section, in order to emulate the noisy implementation of the quantized (finite-precision) MS decoder.

1) *Noisy adder model:* We consider a  $\theta$ -bit adder, with  $\theta \geq 2$ . The inputs and the output of the adder are assumed to be in  $\mathcal{V} = \{-\Theta, \dots, -1, 0, +1, \dots, +\Theta\}$ , where  $\Theta = 2^{\theta-1} - 1$ .

We denote by  $s_{\mathcal{V}} : \mathbb{Z} \rightarrow \mathcal{V}$ , the  $\theta$ -bit saturation map, defined by:

$$s_{\mathcal{V}}(v) = \begin{cases} -\Theta, & \text{if } v < -\Theta \\ v, & \text{if } v \in \mathcal{V} \\ +\Theta, & \text{if } v > +\Theta \end{cases} \quad (8)$$

For inputs  $(x, y) \in \mathcal{V}$ , the output of the noiseless adder is defined as  $s_{\mathcal{V}}(x + y)$ . Hence, for a given error injection model  $(\mathcal{E}, p_{\mathcal{E}}, \iota | \mathcal{V})$ , the output of the noisy adder is given by:

$$\mathbf{a}_{\text{pr}}(x, y) = \iota(s_{\mathcal{V}}(x + y), e), \quad (9)$$

where  $e$  is drawn randomly from  $\mathcal{E}$  according to the probability distribution  $p_{\mathcal{E}}$ . The *error probability of the noisy adder*, assuming uniformly distributed inputs, is equal to the error injection probability (parameter  $p_0$  defined in (3)), and will be denoted in the sequel by  $p_a$ .

2) *Noisy comparator model*: Let  $\mathbf{lt}$  denote the noiseless *less than* operator, defined by  $\mathbf{lt}(x, y) = 1$  if  $x < y$ , and  $\mathbf{lt}(x, y) = 0$  otherwise. The *noisy less than* operator, denoted by  $\mathbf{lt}_{\text{pr}}$ , is defined by injecting errors on the output of the noiseless one, according to the bit-flipping model defined in Section III-D. In other words, the output of the noiseless  $\mathbf{lt}$  operator is flipped with some probability value, which will be denoted in the sequel by  $p_c$ .

Finally, the *noisy minimum* operator is defined by:

$$\mathbf{m}_{\text{pr}}(x, y) = \begin{cases} x, & \text{if } \mathbf{lt}_{\text{pr}}(x, y) = 1 \\ y, & \text{if } \mathbf{lt}_{\text{pr}}(x, y) = 0 \end{cases} \quad (10)$$

3) *Noisy XOR model*: The noisy XOR operator, denoted by  $\mathbf{x}_{\text{pr}}$  is defined by flipping the output of the noiseless operator with some probability value, which will be denoted in the sequel by  $p_x$  (according to the bit-flipping error injection model in Section III-D). It follows that:

$$\mathbf{x}_{\text{pr}}(x, y) = \begin{cases} x \wedge y, & \text{with probability } 1 - p_x \\ \overline{x \wedge y}, & \text{with probability } p_x \end{cases} \quad (11)$$

**Assumption:** We further assume that the inputs and the output of the XOR operator may take values in either  $\{0, 1\}$  or  $\{-1, +1\}$  (using the usual 0,1 to  $\pm 1$  conversion). This assumption will be implicitly made throughout the paper.

**Remark:** As a general rule, we shall refer to a noisy operator according to its underlying error injection model. For instance, a sign-preserving (resp. full-depth or sign-preserving bitwise-XORed) noisy adder, is a noisy adder whose underlying error injection model is sign-preserving (resp. one of the bitwise-XOR error injection models defined in Section III-C). We shall also say that a noisy operator is (*highly*) *symmetric* if its underlying error injection model is so.

### F. Nested Operators

As it can be observed from Algorithm 1, several arithmetic/logic operations must be nested<sup>2</sup> in order to compute the exchanged messages. Since all these operations (additions, comparisons, XOR) are commutative, the way they are nested does not have any impact on the infinite-precision MS decoding. However, this is no longer true for finite-precision decoding, especially in case of noisy operations. Therefore, one needs an assumption about how operators extend from two to more inputs.

Our assumption is the following. For  $n \geq 2$  inputs, we randomly pick any two inputs and apply the operator on this pair. Then we replace the pair by the obtained output, and repeat the above procedure until there is only one output (and no more inputs) left.

The formal definition goes as follows. Let  $\Omega \subset \mathbb{Z}$  and  $\omega : \Omega \times \Omega \rightarrow \Omega$  be a noiseless or noisy operator with two operands. Let  $\{x_i\}_{i=1:n} \subset \Omega$  be an unordered set of  $n$  operands. We define:

$$\omega(\{x_i\}_{i=1:n}) = \omega(\dots(\omega(x_{\pi(1)}, x_{\pi(2)}), \dots), x_{\pi(n)}),$$

where  $\pi$  is a random permutation of  $1, \dots, n$ .

## IV. NOISY MIN-SUM DECODING

### A. Finite-Precision Min-Sum Decoder

We consider a finite-precision MS decoder, in which the a priori information ( $\gamma_n$ ) and the exchanged messages ( $\alpha_{m,n}$  and  $\beta_{m,n}$ ) are quantized on  $q$  bits. The a posteriori information ( $\tilde{\gamma}_n$ ) is quantized on  $\tilde{q}$  bits, with  $\tilde{q} > q$  (usually  $\tilde{q} = q + 1$ , or  $\tilde{q} = q + 2$ ). We further denote:

- $\mathcal{M} = \{-Q, \dots, -1, 0, +1, \dots, Q\}$ , where  $Q = 2^{q-1} - 1$ , the alphabet of both the a priori information and the exchanged messages;
- $\tilde{\mathcal{M}} = \{-\tilde{Q}, \dots, -1, 0, +1, \dots, \tilde{Q}\}$ , where  $\tilde{Q} = 2^{\tilde{q}-1} - 1$ , the alphabet of the a posteriori information;
- $\mathbf{q} : \mathcal{Y} \rightarrow \mathcal{M}$ , a quantization map, where  $\mathcal{Y}$  denotes the channel output alphabet;
- $\mathbf{s}_{\mathcal{M}} : \mathbb{Z} \rightarrow \mathcal{M}$ , the  $q$ -bit saturation map (defined in a similar manner as in (8));
- $\mathbf{s}_{\tilde{\mathcal{M}}} : \mathbb{Z} \rightarrow \tilde{\mathcal{M}}$ , the  $\tilde{q}$ -bit saturation map

**Remark:** The quantization map  $\mathbf{q}$  determines the  $q$ -bit quantization of the decoder soft input. Since  $\mathbf{q}$  is defined on the channel input (*i.e.*  $y_n$  values), it must also encompass the computation of the corresponding LLR values, whenever is necessary (see also the Remark at the end of Section II-B).

<sup>2</sup>For instance,  $(d_n - 1)$  additions – where  $d_n$  denotes the degree of the variable-node  $n$  – are required in order to compute each  $\alpha_{m,n}$  message. Similarly, each  $\beta_{m,n}$  message requires  $(d_m - 2)$  XOR operations and  $(d_m - 2)$  comparisons.

Saturation maps  $s_{\mathcal{M}}$  and  $s_{\widetilde{\mathcal{M}}}$  define the finite-precision saturation of the exchanged messages and of the a posteriori information, respectively.

### B. Noisy Min-Sum Decoder

The noisy (finite-precision) MS decoding is presented in Algorithm 2. We assume that  $\tilde{q}$ -bit adders are used to compute both  $\alpha_{m,n}$  messages in the **VN-processing** step, and  $\tilde{\gamma}_n$  values in the **AP-update** processing step. This is usually the case in practical implementations<sup>3</sup>, and allows us to use the same type of adder in both processing steps. This assumption explains as well the  $q$ -bit saturation of  $\alpha_{m,n}$  messages in the **VN-processing** step. Note also that the saturation of  $\tilde{\gamma}_n$  values is actually done within the adder (see Equation (9)).

Finally, we note that the *hard decision* and the *syndrome check* steps in Algorithm 2 are assumed to be *noiseless*. We note however that the syndrome check step is optional, and if missing, the decoder stops when the maximum number of iterations is reached.

### C. Sign-Preserving Properties

Let  $\mathbf{U}$  denote any of the VN-processing or CN-processing units of the noiseless MS decoder. We denote by  $\mathbf{U}_{\text{pr}}$  the corresponding unit of the noisy MS decoder. We say that  $\mathbf{U}_{\text{pr}}$  is *sign-preserving* if for any incoming messages and any noise realization, the outgoing message is of the same sign as the message obtained when the same incoming messages are supplied to  $\mathbf{U}$ .

Clearly,  $\text{CN}_{\text{pr}}$  is sign-preserving if and only if the XOR-operator is noiseless ( $p_x = 0$ ). In case that the noisy XOR-operator severely degrades the decoder performance, it is possible to increase its reliability by using classical fault-tolerant techniques (as for instance modular redundancy, or multi-voltage design by increasing the supply voltage of the corresponding XOR-gate). The price to pay, when compared to the size or the energy consumption of the whole circuit, would be reasonable.

Concerning the VN-processing, it is worth noting that the  $\text{VN}_{\text{pr}}$  is **not** sign-preserving, even if the noisy adder is. This is due to the fact that multiple adders must be nested in order to complete the VN-processing. However, a sign-preserving adder might have several benefits. First, the error probability of the sign of variable-node messages would be lowered, which would certainly help the decoder. Second, if the noisy adder is sign-preserving and all the variable-node incoming messages have the same sign, then

<sup>3</sup>In practical implementation, the  $\tilde{\gamma}_n$  is computed first, and then  $\alpha_{m,n}$  is obtained from  $\tilde{\gamma}_n$  by subtracting the incoming  $\beta_{m,n}$  message

---

**Algorithm 2** Noisy Min-Sum (Noisy-MS) decoding
 

---

 Input:  $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$  ( $\mathcal{Y}$  is the channel output alphabet) ▷ received word

 Output:  $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{-1, +1\}^N$  ▷ estimated codeword
**Initialization**
**for all**  $n = 1, \dots, N$  **do**  $\gamma_n = \mathbf{q}(y_n)$ ;

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do**  $\alpha_{m,n} = \gamma_n$ ;
**Iteration Loop**
**for all**  $m = 1, \dots, M$  and  $n \in \mathcal{H}(m)$  **do** ▷ **CN-processing**

$$\beta_{m,n} = \mathbf{x}_{\text{pr}}(\{\text{sgn}(\alpha_{m,n'})\}_{n' \in \mathcal{H}(m) \setminus n}) \mathbf{m}_{\text{pr}}(\{|\alpha_{m,n'}|\}_{n' \in \mathcal{H}(m) \setminus n});$$

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do** ▷ **VN-processing**

$$\alpha_{m,n} = \mathbf{a}_{\text{pr}}(\{\gamma_n\} \cup \{\beta_{m',n}\}_{m' \in \mathcal{H}(n) \setminus m});$$

$$\alpha_{m,n} = \mathbf{s}_{\mathcal{M}}(\alpha_{m,n});$$

**for all**  $n = 1, \dots, N$  **do** ▷ **AP-update**

$$\tilde{\gamma}_n = \mathbf{a}_{\text{pr}}(\{\gamma_n\} \cup \{\beta_{m,n}\}_{m \in \mathcal{H}(n)});$$

**for all**  $\{v_n\}_{n=1, \dots, N}$  **do**  $\hat{x}_n = \text{sgn}(\tilde{\gamma}_n)$ ; ▷ hard decision
**if**  $\hat{\underline{x}}$  is a codeword **then** exit the iteration loop ▷ syndrome check
**End Iteration Loop**


---

the  $\text{VN}_{\text{pr}}$  does preserve the sign of the outgoing message. Put differently, in case that all the incoming messages agree on the same hard decision, the noisy VN-processing may change the confidence level, but cannot change the decision. This may be particularly useful, especially during the last decoding iterations.

Finally, the motivation behind the sign-preserving noisy adder model is to investigate its possible benefits on the decoder performance. If the benefits are worth it (*e.g.* one can ensure a target performance of the decoder), the sign-bit of the adder could be protected by using classical fault-tolerant solutions.

## V. DENSITY EVOLUTION

### A. Concentration and Convergence Properties

First, we note that our definition of *symmetry* is slightly more general than the one used in [9]. Indeed, even if all the error injection models used within the noisy MS decoder are *symmetric*, the noisy MS decoder does not necessarily verify the **symmetry** property from [9]. However, this property is verified

in case of *highly symmetric* fault injection<sup>4</sup>. Nevertheless, the concentration and convergence properties proved in [9] for **symmetric** noisy message-passing decoders, can easily be generalized to our definition of *symmetry*.

We summarize below the most important results; the proof relies essentially on the same arguments as in [9]. We consider an *ensemble* of LDPC codes, with length  $N$  and fixed degree distribution polynomials [24]. We choose a random code  $\mathbf{C}$  from this ensemble and assume that a random codeword  $\mathbf{x} \in \{-1, +1\}^N$  is sent over a binary-input memoryless symmetric channel. We fix some number of decoding iterations  $\ell > 0$ , and denote by  $E_{\mathbf{C}}^{(\ell)}(\mathbf{x})$  the expected fraction of incorrect *messages*<sup>5</sup> at iteration  $\ell$ .

*Theorem 1:* Assume that all the error injection models used within the MS decoder are symmetric. Then, the following properties hold:

- 1) [*Conditional Independence of Error*] For any decoding iteration  $\ell > 0$ , the expected fraction of incorrect messages  $E_{\mathbf{C}}^{(\ell)}(\mathbf{x})$  does not depend on  $\mathbf{x}$ . Therefore, we may define  $E_{\mathbf{C}}^{(\ell)} := E_{\mathbf{C}}^{(\ell)}(\mathbf{x})$ .
- 2) [*Cycle-Free Case*] If the graph of  $\mathbf{C}$  contains no cycles of length  $2\ell$  or less, the expected fraction of incorrect messages  $E_{\mathbf{C}}^{(\ell)}$  does not depend on the code  $\mathbf{C}$  or the code-length  $N$ , but only on the degree distribution polynomials; in this case, it will be further denoted by  $E_{\infty}^{(\ell)}(\mathbf{x})$ .
- 3) [*Concentration Around the Cycle-Free Case*] For any  $\delta > 0$ , the probability that  $E_{\mathbf{C}}^{(\ell)}$  lies outside the interval  $(E_{\infty}^{(\ell)}(\mathbf{x}) - \delta, E_{\infty}^{(\ell)}(\mathbf{x}) + \delta)$  converges to zero exponentially fast in  $N$ .

## B. Density Evolution Equations

In this section we derive density evolution equations for the noisy finite-precision MS decoding for a regular  $(d_v, d_c)$  LDPC code. The study can be easily generalized to irregular LDPC codes, simply by averaging according to the degree distribution polynomials.

The objective of the density evolution technique is to recursively compute the probability mass functions of exchanged messages, through the iterative decoding process. This is done under the independence assumption of exchanged messages, holding in the asymptotic limit of the code length, in which case the decoding performance converges to the cycle-free case. Due to the symmetry of the decoder, the analysis

<sup>4</sup>According to the probabilistic models introduced in Section III-E, the noisy comparator and the noisy XOR-operator are highly symmetric, but the noisy adder does not necessarily be so!

<sup>5</sup>Here, “*messages*” may have any one of the three following meanings: “variable-node messages”, or “check-node messages”, or “a posteriori information values”.

can be further simplified by assuming that the all-zero codeword is transmitted through the channel. We note that our analysis applies to any memoryless symmetric channel.

Let  $\ell > 0$  denote the decoding iteration. Superscript  $(\ell)$  will be used to indicate the messages and the a posteriori information computed at iteration  $\ell$ . To indicate the value of a message on a randomly selected edge, we drop the variable and check node indexes from the notation (and we proceed in a similar manner for the a priori and a posteriori information). The corresponding probability mass functions are denoted as follows.

$$\begin{aligned} C(z) &= \Pr(\gamma = z), & \forall z \in \mathcal{M} \\ \tilde{C}^{(\ell)}(\tilde{z}) &= \Pr(\tilde{\gamma}^{(\ell)} = \tilde{z}), & \forall \tilde{z} \in \tilde{\mathcal{M}} \\ A^{(\ell)}(z) &= \Pr(\alpha^{(\ell)} = z), & \forall z \in \mathcal{M} \\ B^{(\ell)}(z) &= \Pr(\beta^{(\ell)} = z), & \forall z \in \mathcal{M} \end{aligned}$$

1) *Expression of the input probability mass function  $C$* : The probability mass function  $C$  depends only on the channel and the quantization map  $\mathbf{q} : \mathcal{Y} \rightarrow \mathcal{M}$ , where  $\mathcal{Y}$  denotes the channel output alphabet (Section IV-A). We also note that for  $\ell = 0$ , we have  $A^{(0)} = C$ .

We give below the expression of  $C$  for the BSC and the BI-AWGN channel models (see Section II-B). For the BSC, the channel output alphabet is  $\mathcal{Y} = \{-1, +1\}$ , while for the BI-AWGN channel,  $\mathcal{Y} = \mathbb{R}$ .

Let  $\mu$  be a positive number, such that  $\mu \leq Q$ . The quantization map  $\mathbf{q}_\mu$  is defined as follows:

$$\mathbf{q}_\mu : \mathcal{Y} \rightarrow \mathcal{M}, \quad \mathbf{q}_\mu(y) = \mathbf{s}_{\mathcal{M}}([\mu \cdot y]), \quad (12)$$

where  $[\mu \cdot y]$  denotes the nearest integer to  $\mu \cdot y$ , and  $\mathbf{s}_{\mathcal{M}}$  is the saturation map (Section IV-A). For the BSC, we will further assume that  $\mu$  is an integer. It follows that  $\mathbf{q}_\mu(y) = \mu \cdot y, \forall y \in \mathcal{Y} = \{-1, +1\}$ .

Considering the all-zero (+1) codeword assumption, the probability mass function  $C$  can be computed as follows.

- For the BSC with crossover probability  $\varepsilon$ :

$$C(z) = \begin{cases} 1 - \varepsilon, & \text{if } z = \mu \\ \varepsilon, & \text{if } z = -\mu \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

- For the BI-AWGN channel with noise variance  $\sigma^2$ :

$$C(z) = \begin{cases} 1 - q\left(\frac{-Q+0.5-\mu}{\mu\sigma}\right), & \text{if } z = -Q \\ q\left(\frac{z-0.5-\mu}{\mu\sigma}\right) - q\left(\frac{z+0.5-\mu}{\mu\sigma}\right), & \text{if } -Q < z < +Q \\ q\left(\frac{Q-0.5-\mu}{\mu\sigma}\right), & \text{if } z = +Q \end{cases} \quad (14)$$



where  $q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} \exp\left(-\frac{u^2}{2}\right) du$  is the tail probability of the standard normal distribution (also known as the *Q-function*).

2) *Expression of  $B^{(\ell)}$  as a function of  $A^{(\ell-1)}$ :*

In the sequel, we make the convention that  $\Pr(\text{sgn}(0) = 1) = \Pr(\text{sgn}(0) = -1) = 1/2$ . The following notation will be used:

- $A_{[x,y]} = \sum_{z=x}^y A(z)$ , for  $x \leq y \in \mathcal{M}$
- $A_{[0^+,y]} = \frac{1}{2}A(0) + \sum_{z=1}^y A(z)$ , for  $y \in \mathcal{M}$ ,  $y > 0$
- $A_{[x,0^-]} = \frac{1}{2}A(0) + \sum_{z=x}^{-1} A(z)$ , for  $x \in \mathcal{M}$ ,  $x < 0$

For the sake of simplicity, we drop the iteration index, thus  $B := B^{(\ell)}$  and  $A := A^{(\ell-1)}$ . We proceed by recursion on  $i = 2, \dots, d_c - 1$ , where  $d_c$  denotes the check-node degree.

Let  $\beta_1 := \alpha_1$ , and for  $i = 2, \dots, d_c - 1$  define:

$$\beta_i = \mathbf{x}_{\text{pr}}(\text{sgn}(\beta_{i-1}), \text{sgn}(\alpha_i)) \mathbf{m}_{\text{pr}}(|\beta_{i-1}|, |\alpha_i|)$$

Let also  $B_{i-1}$  and  $B_i$  denote the probability mass functions of  $\beta_{i-1}$  and  $\beta_i$ , respectively (hence,  $B_1 = A$ ).

First of all, for  $z = 0$ , we have:

$$B_i(0) = \Pr(\beta_i = 0) = A(0)B_{i-1}(0) + [B_{i-1}(0)(1 - A(0)) + A(0)(1 - B_{i-1}(0))] (1 - p_c).$$

For  $z \neq 0$ , we proceed in several steps as follows:

For  $z > 0$ :

$$\begin{aligned} F'_i(z) &\stackrel{\text{def}}{=} \Pr(\beta_i \geq z \mid p_x = 0) \\ &= \left[ B_{i-1[0^+,z-1]} A_{[z,Q-1]} + A_{[0^+,z-1]} B_{i-1[z,Q-1]} \right] p_c \\ &\quad + \left[ B_{i-1[1-z,0^-]} A_{[-Q,-z]} + A_{[1-z,0^-]} B_{i-1[-Q,-z]} \right] p_c \\ &\quad + B_{i-1[z,Q-1]} A_{[z,Q-1]} + B_{i-1[-Q,-z]} A_{[-Q,-z]} \end{aligned}$$

$$\begin{aligned} F_i(z) &\stackrel{\text{def}}{=} \Pr(\beta_i \geq z) \\ &= (1 - p_x) \cdot F'_i(z) + p_x \cdot G'_i(-z) \end{aligned}$$

$$B_i(z) = \Pr(\beta_i = z) = F_i(z) - F_i(z+1)$$

For  $z < 0$ :

$$\begin{aligned} G'_i(z) &\stackrel{\text{def}}{=} \Pr(\beta_i \leq z \mid p_x = 0) \\ &= \left[ B_{i-1[0^+,-z-1]} A_{[-Q,z]} + A_{[0^+,-z-1]} B_{i-1[-Q,z]} \right] p_c \\ &\quad + \left[ B_{i-1[-z,Q-1]} A_{[z+1,0^-]} + A_{[-z,Q-1]} B_{i-1[z+1,0^-]} \right] p_c \\ &\quad + B_{i-1[-z,Q-1]} A_{[-Q,z]} + A_{[-z,Q-1]} B_{i-1[-Q,z]} \end{aligned}$$

$$\begin{aligned} G_i(z) &\stackrel{\text{def}}{=} \Pr(\beta_i \leq z) \\ &= (1 - p_x) \cdot G'_i(z) + p_x \cdot F'_i(-z) \end{aligned}$$

$$B_i(z) = \Pr(\beta_i = z) = G_i(z) - G_i(z+1)$$

Finally, we have that  $B = B_{d_c-1}$ .

3) *Expression of  $A^{(\ell)}$  as a function of  $B^{(\ell)}$  and  $C$ :* We derive at the same time the expression of  $\tilde{C}^{(\ell)}$  as a function of  $B^{(\ell)}$  and  $C$ .

For simplicity, we drop the iteration index, so  $A := A^{(\ell)}$ ,  $B := B^{(\ell)}$ , and  $\tilde{C} := \tilde{C}^{(\ell)}$ . We denote by  $(\mathcal{E}, p_{\mathcal{E}}, \iota \mid \tilde{\mathcal{M}})$  the error injection model used to define the noisy adder. We decompose each noisy

addition into three steps (noiseless infinite-precision addition, saturation, and error injection), and proceed by recursion on  $i = 0, 1, \dots, d_v$ , where  $d_v$  denotes the variable-node degree:

- For  $i = 0$ :

$$\Omega_0 \stackrel{\text{def}}{=} \gamma \in \mathcal{M} \subseteq \widetilde{\mathcal{M}}, \quad \widetilde{C}_0(\tilde{z}) \stackrel{\text{def}}{=} \Pr(\Omega_0 = \tilde{z}) = \begin{cases} C(\tilde{z}), & \text{if } \tilde{z} \in \mathcal{M} \\ 0, & \text{if } \tilde{z} \in \widetilde{\mathcal{M}} \setminus \mathcal{M} \end{cases}$$

- For  $i = 1, \dots, d_v$ :

$$\omega_i \stackrel{\text{def}}{=} \Omega_{i-1} + \beta_{m_i, n} \in \mathbb{Z}, \quad c_i(w) \stackrel{\text{def}}{=} \Pr(\omega_i = w) = \sum_u \widetilde{C}_{i-1}(u) B(w - u), \forall w \in \mathbb{Z}$$

$$\tilde{\omega}_i \stackrel{\text{def}}{=} \mathbf{s}_{\widetilde{\mathcal{M}}}(\omega_i) \in \widetilde{\mathcal{M}}, \quad \tilde{c}_i(\tilde{w}) \stackrel{\text{def}}{=} \Pr(\tilde{\omega}_i = \tilde{w}) = \begin{cases} c_i(\tilde{w}), & \text{if } \tilde{w} \in \widetilde{\mathcal{M}} \setminus \{\pm \tilde{Q}\} \\ \sum_{w \leq -\tilde{Q}} c_i(w), & \text{if } \tilde{w} = -\tilde{Q} \\ \sum_{w \geq +\tilde{Q}} c_i(w), & \text{if } \tilde{w} = +\tilde{Q} \end{cases}$$

$$\Omega_i \stackrel{\text{def}}{=} \iota(\tilde{\omega}_i, e) \in \widetilde{\mathcal{M}}, \quad \widetilde{C}_i(\tilde{z}) \stackrel{\text{def}}{=} \Pr(\Omega_i = \tilde{z}) = \sum_{\tilde{\omega}} \sum_e \delta_{\iota(\tilde{\omega}, e)}^{\tilde{z}} p_{\mathcal{E}}(e) \tilde{c}_i(\tilde{\omega}), \forall \tilde{z} \in \widetilde{\mathcal{M}}$$

where  $\delta_x^y = 1$  if  $x = y$ , and  $\delta_x^y = 0$  if  $x \neq y$ .

Note that in the definition of  $\Omega_i$  above,  $e$  denotes an error drawn from the error set  $\mathcal{E}$  according to the error probability distribution  $p_{\mathcal{E}}$ .

Finally, we have:

- $A = \mathbf{s}_{\mathcal{M}}(\widetilde{C}_{d_v-1})$
- $\widetilde{C} = \widetilde{C}_{d_v}$

In the first equation above, applying the saturation operator  $\mathbf{s}_{\mathcal{M}}$  on the probability mass function  $\widetilde{C}_{d_v-1}$  means that all the probability weights corresponding to values  $\tilde{w}$  outside  $\mathcal{M}$  must be accumulated to the probability of the corresponding boundary value of  $\mathcal{M}$  (that is, either  $-Q$  or  $+Q$ , according to whether  $\tilde{w} < -Q$  or  $\tilde{w} > +Q$ ).

**Remark:** If the noisy adder is defined by one of the bitwise-XOR error injection models (Section III-C), then the third equation from the above recursion (expression of  $\widetilde{C}_i$  as a function of  $\tilde{c}_i$ ) may be rewritten as follows:

- Sign-preserving bitwise-XORed noisy adder

$$\widetilde{C}_i(\tilde{z}) = \begin{cases} (1 - p_a) \tilde{c}_i(\tilde{z}) + \frac{1}{\tilde{Q}} p_a (\tilde{c}_i[\leq 0^-] - \tilde{c}_i(\tilde{z})), & \text{if } \tilde{z} < 0 \\ (1 - p_a) \tilde{c}_i(0) + \frac{1}{\tilde{Q}} p_a (1 - \tilde{c}_i(0)), & \text{if } \tilde{z} = 0 \\ (1 - p_a) \tilde{c}_i(\tilde{z}) + \frac{1}{\tilde{Q}} p_a (\tilde{c}_i[\geq 0^+] - \tilde{c}_i(\tilde{z})), & \text{if } \tilde{z} > 0 \end{cases} \quad (15)$$

where  $\tilde{c}_{i[\leq 0^-]} = \sum_{\tilde{\omega} < 0} \tilde{c}_i(\tilde{\omega}) + \frac{1}{2}\tilde{c}_i(0)$ , and  $\tilde{c}_{i[\geq 0^+]} = \frac{1}{2}\tilde{c}_i(0) + \sum_{\tilde{\omega} > 0} \tilde{c}_i(\tilde{\omega})$ .

- Full-depth bitwise-XORed noisy adder

$$\tilde{C}_i(\tilde{z}) = (1 - p_a)\tilde{c}_i(\tilde{z}) + \frac{1}{2\tilde{Q}}p_a(1 - \tilde{c}_i(\tilde{z})) \quad (16)$$

Finally, we note that the density evolution equations for the noiseless finite-precision MS decoder can be obtained by setting  $p_a = p_c = p_x = 0$ .

### C. Error Probability and Useful and Target-BER regions

- 1) *Decoding Error Probability*: The error probability at decoding iteration  $\ell$ , is defined by:

$$P_e^{(\ell)} = \sum_{\tilde{z}=-\tilde{Q}}^{-1} \tilde{C}^{(\ell)}(\tilde{z}) + \frac{\tilde{C}^{(\ell)}(0)}{2} \quad (17)$$

*Proposition 1*: The error probability at decoding iteration  $\ell$  is lower-bounded as follows:

- (a) For the sign-preserving bitwise-XORed noisy adder:  $P_e^{(\ell)} \geq \frac{1}{2\tilde{Q}}p_a$ .
- (b) For the full-depth bitwise-XORed noisy adder:  $P_e^{(\ell)} \geq \frac{1}{2}p_a + \frac{1}{4\tilde{Q}}p_a$ .

*Proof.* (a) Using  $\tilde{C} = \tilde{C}_{d_v}$  and equations (17) and (15), it follows that  $P_e^{(\ell)} = (1 - p_a)\tilde{c}_{d_v[\leq 0^-]} + \frac{1}{2\tilde{Q}}p_a(1 - 2\tilde{c}_{d_v[\leq 0^-]}) + p_a(\tilde{c}_{d_v[\leq 0^-]} - \frac{1}{2}\tilde{c}_{d_v}(0)) \geq (1 - p_a)\tilde{c}_{d_v[\leq 0^-]} + \frac{1}{2\tilde{Q}}p_a(1 - 2\tilde{c}_{d_v[\leq 0^-]}) \geq \frac{1}{2\tilde{Q}}p_a$ , since the function  $(1 - p_a)x + \frac{1}{2\tilde{Q}}p_a(1 - 2x)$  is an increasing function of  $x \in [0, 1]$ .

(b) Equations (17) and (16) imply that  $P_e^{(\ell)} = \frac{1}{2}p_a + (1 - p_a)\tilde{c}_{d_v[\leq 0^-]} + \frac{1}{4\tilde{Q}}p_a(1 - 2\tilde{c}_{d_v[\leq 0^-]}) \geq \frac{1}{2}p_a + \frac{1}{4\tilde{Q}}p_a$   $\square$

Note that the above lower bounds are actually inferred from the error injection in the *last (the  $d_v$ -th) addition* performed when computing the a posteriori information value. Therefore, these lower bounds are not expected to be tight. However, if the channel error probability is small enough, the sign-preserving lower bound proves to be tight in the asymptotic limit of  $\ell$  (this will be discussed in more details in Section VI). Note also that by protecting the sign of the noisy adder, the bound is lowered by a factor of roughly  $\tilde{Q}$ , which represents an exponential improvement with respect to the number of bits of the adder.

In the asymptotic limit of the code-length,  $P_e^{(\ell)}$  gives the probability of the hard bit estimates being in error at decoding iteration  $\ell$ . For the (noiseless, infinite-precision) BP decoder, the error probability is usually a decreasing function of  $\ell$ . This is no longer true for the noiseless, infinite-precision MS decoder, for which the error probability may increase with  $\ell$ . However, both decoders exhibit a *threshold*

*phenomenon*, separating the region where error probability goes to zero (as the number of decoding iterations goes to infinity), from that where it is bounded above zero [24].

Things get more complicated for the noisy (finite-precision) MS decoder. First, the error probability have a more unpredictable behavior. It does not always converge and it may become periodic<sup>6</sup> when the number of iterations goes to infinity. Second, the error probability is always bounded above zero (Proposition 1), since there is a non-zero probability of fault injection at any decoding iteration. Hence, a decoding threshold, similar to the noiseless case, cannot longer be defined.

Following [9], we define below the notions of useful decoder and target error rate threshold. We consider a channel model depending on a channel parameter  $\chi$ , such that the channel is degraded by increasing  $\chi$  (for example, the crossover probability for the BSC, or the noise variance for the BI-AWGN channel). We will use subscript  $\chi$  to indicate a quantity that depends on  $\chi$ . Hence, in order to account that  $P_e^{(\ell)}$  depends also on the value of the channel parameter, it will be denoted in the following by  $P_{e,\chi}^{(\ell)}$ .

2) *Useful Region*: The first step is to evaluate the channel and hardware parameters yielding a final probability of error (in the asymptotic limit of the number of iterations) less than the *input error probability*. The latter probability is given by  $P_{e,\chi}^{(0)} = \sum_{z=-Q}^{-1} C(z) + \frac{1}{2}C(0)$ , where  $C$  is the probability mass function of the quantized a priori information of the decoder (see Section V-B1).

Following [9], the decoder is said to be *useful* if  $\left(P_{e,\chi}^{(\ell)}\right)_{\ell>0}$  is convergent, and:

$$P_{e,\chi}^{(\infty)} \stackrel{\text{def}}{=} \lim_{\ell \rightarrow \infty} P_{e,\chi}^{(\ell)} < P_{e,\chi}^{(0)} \quad (18)$$

The ensemble of the parameters that satisfy this condition constitutes the *useful region* of the decoder.

3) *Target Error Rate Threshold*: For noiseless-decoders, the decoding threshold is defined as the supremum channel noise, such that the error probability converges to zero as the number of decoding iterations goes to infinity. However, for noisy decoders this error probability does not converge to zero, and an alternative definition of the decoding threshold has been introduced in [9]. Accordingly, for a target bit-error rate  $\eta$ , the  $\eta$ -threshold is defined<sup>7</sup> by:

$$\chi^*(\eta) = \sup \left\{ \chi \mid P_{e,\chi'}^{(\infty)} \text{ exists and } P_{e,\chi'}^{(\infty)} < \eta, \forall \chi' \in [0, \chi] \right\} \quad (19)$$

<sup>6</sup>In fact, for both BSC and BI-AWGN channels, the only cases we observed, in which the sequence  $\left(P_e^{(\ell)}\right)_{\ell>0}$  does not converge, are those cases in which this sequence becomes periodic for  $\ell$  large enough.

<sup>7</sup>In [9], the  $\eta$ -threshold is defined by  $\chi^*(\eta) = \sup \left\{ \chi \mid P_{e,\chi}^{(\infty)} \text{ exists and } P_{e,\chi}^{(\infty)} < \eta \right\}$ , and consequently, there might exist a channel parameter value  $\chi' < \chi^*(\eta)$ , for which  $P_{e,\chi'}^{(\infty)}$  does not exist. In order to avoid this happening, our definition is slightly different from the one in [9].

#### D. Functional Threshold

Although the  $\eta$ -threshold definition allows determining the maximum channel noise for which the bit error probability can be reduced below a target value, there is not significant change in the behavior of the decoder when the channel noise parameter  $\lambda$  increases beyond the value of  $\chi^*(\eta)$ . In this section, a new threshold definition is introduced in order to identify the channel and hardware parameters yielding to a sharp change in the decoder behavior, similar to the change that occurs around the threshold of the noiseless decoder. This threshold will be referred to as the *functional threshold*. The aim is to detect a sharp increase (e.g. discontinuity) in the error probability of the noisy decoder, when  $\lambda$  goes beyond this functional threshold value. The threshold definition we propose make use of the Lipschitz constant of the function  $\chi \mapsto P_e^{(\infty)}(\chi)$  in order to detect a sharp change of  $P_e^{(\infty)}(\chi)$  with respect to  $\chi$ . The definition of the Lipschitz constant is first restated for the sake of clarity.

*Definition 4:* Let  $f : I \rightarrow \mathbb{R}$  be a function defined on an interval  $I \subseteq \mathbb{R}$ . The *Lipschitz constant* of  $f$  in  $I$  is defined as

$$L(f, I) = \sup_{x \neq y \in I} \frac{|f(x) - f(y)|}{|x - y|} \in \mathbb{R}_+ \cup \{+\infty\} \quad (20)$$

For  $a \in I$  and  $\delta > 0$ , let  $I_a(\delta) = I \cap (a - \delta, a + \delta)$ . The *(local) Lipschitz constant* of  $f$  in  $a \in I$  is defined by:

$$L(f, a) = \inf_{\delta > 0} L(f, I_a(\delta)) \in \mathbb{R}_+ \cup \{+\infty\} \quad (21)$$

Note that if  $a$  is a discontinuity point of  $f$ , then  $L(f, a) = +\infty$ . On the opposite, if  $f$  is differentiable in  $a$ , then the Lipschitz constant in  $a$  corresponds to the absolute value of the derivative. Furthermore, if  $L(f, I) < +\infty$ , then  $f$  is uniformly continuous on  $I$  and almost everywhere differentiable. In this case,  $f$  is said to be *Lipschitz continuous* on  $I$ .

The functional threshold is then defined as follows.

*Definition 5:* For given hardware parameters and a channel parameter  $\chi$ , the decoder is said to be *functional* if

- (a) The function  $x \mapsto P_e^{(\infty)}(x)$  is defined on  $[0, \chi]$
- (b)  $P_e^{(\infty)}$  is Lipschitz continuous on  $[0, \chi]$
- (c)  $L(P_e^{(\infty)}, x)$  is an increasing function of  $x \in [0, \chi]$

Then, the functional threshold  $\bar{\chi}$  is defined as:

$$\bar{\chi} = \sup\{\chi \mid \text{conditions (a), (b) and (c) are satisfied}\} \quad (22)$$

The use of the Lipschitz constant allows a rigorous definition of the functional threshold, while avoiding the use of the derivative (which would require  $P_e^{(\infty)}(\lambda)$  to be a piecewise differentiable function of  $\lambda$ ). As it will be further illustrated in Section VI, the functional threshold corresponds to a transition between two modes. The first mode corresponds to the channel parameters leading to a low level of error probability, *i.e.*, for which the decoder can correct most of the errors from the channel. In the second mode, the channel parameters lead to a much higher error probability level. If  $L\left(P_e^{(\infty)}, \bar{\chi}\right) = +\infty$ , then  $\bar{\chi}$  is a discontinuity point of  $P_e^{(\infty)}$  and the transition between the two levels is sharp. If  $L\left(P_e^{(\infty)}, \bar{\chi}\right) < +\infty$ , then  $\bar{\chi}$  is an inflection point of  $P_e^{(\infty)}$  and the transition is smooth. With the Lipschitz constant, one can characterize the transition in both cases. However, the second case corresponds to a degenerated one, in which the hardware noise is too high and leads to a non-standard asymptotic behavior of the decoder. That is why a set of admissible hardware noise parameters is defined as follows.

*Definition 6:* The set of *admissible hardware parameters* is the set of hardware noise parameters  $(p_a, p_c, p_x)$  for which  $L\left(P_e^{(\infty)}, \bar{\chi}\right) = +\infty$ .

In the following, as each threshold definition helps at illustrating different effects, one or the other definition will be used, depending on the context.

## VI. ASYMPTOTIC ANALYSIS OF THE NOISY MIN-SUM DECODER

In this section, the density evolution equations derived previously are used to analyze the asymptotic performance (*i.e.* in the asymptotic limit of both the code length and number of iterations) of the noisy MS decoder.

Unless specified otherwise, the following parameters are used throughout this section:

### Code parameters:

- We consider the ensemble of regular LDPC codes with variable-node degree  $d_v = 3$  and check-node degree  $d_c = 6$

### Quantization parameters:

- The a priori information and exchanged messages are quantized on  $q = 4$  bits; hence,  $Q = 7$  and  $\mathcal{M} = \{-7, \dots, +7\}$ .
- The a posteriori information is quantized on  $\tilde{q} = 5$  bits; hence,  $\tilde{Q} = 15$  and  $\tilde{\mathcal{M}} = \{-15, \dots, +15\}$ .

We analyze the decoding performance depending on:

- The quantization map  $\mathbf{q}_\mu : \mathcal{Y} \rightarrow \mathcal{M}$ , defined in Equation (12). The factor  $\mu$  will be referred to as the *channel-output scale factor*, or simply the *channel scale factor*.

- The parameters of the noisy adder, comparator, and XOR-operator, defined respectively in Equations (9), (10), and (11).

#### A. Numerical results for the BSC

For the BSC, the channel output alphabet is  $\mathcal{Y} = \{-1, +1\}$  and the quantization map is defined by  $\mathbf{q}_\mu(-1) = -\mu$  and  $\mathbf{q}_\mu(+1) = +\mu$ , with  $\mu \in \{1, \dots, Q\}$ .

The infinite-precision MS decoder (Algorithm 1), is known to be independent of the scale factor  $\mu$ . This is because  $\mu$  factors out from all the processing steps in Algorithm 1, and therefore does not affect in any way the decoding process. This is no longer true for the finite precision decoder (due to saturation effects), and we will show in this section that, even in the noiseless case, the scale factor  $\mu$  can significantly impact the performance of the finite precision MS decoder.

We start by analyzing the performance of the MS decoder with quantization map  $\mathbf{q}_1$ , and then we will analyze its performance with an optimized quantization map  $\mathbf{q}_\mu$ .

1) *Min-Sum decoder with quantization map  $\mathbf{q}_1$* : The case  $\mu = 1$  leads to an “unconventional” behavior, as in some particular cases the noise introduced by the device can help the MS decoder to escape from fixed points attractors, and may actually result in an increased correction capacity with respect to the noiseless decoder. This behavior will be discussed in more details in this section.

We start with the noiseless decoder case. Figure 1 shows the asymptotic error probability  $P_e^{(\infty)}$  as a function of  $p_0$ . It can be seen that  $P_e^{(\infty)}$  decreases slightly with  $p_0$ , until  $p_0$  reaches a threshold value  $p_{\text{th}} = 0.039$ , where  $P_e^{(\infty)}$  drops to zero. This is the *classical* threshold phenomenon mentioned in Section V-C: for  $p_0 > p_{\text{th}}$ , the decoding error probability is bounded far above zero ( $P_e^{(\infty)} > 0.31$ ), while for  $p_0 < p_{\text{th}}$ , one has  $P_e^{(\infty)} = 0$ .

Now, we consider a  $p_0$  value slightly greater than the threshold of the noiseless decoder, and investigate the effect of the noisy adder on the decoder performance. Let us fix  $p_0 = 0.06$ . Figure 2(a) shows the decoding error probability at iteration  $\ell$ , for different parameters  $p_a \in \{10^{-30}, 10^{-15}, 10^{-5}\}$  of the noisy adder. For each  $p_a$  value, there are two superimposed curves, corresponding to the full-depth (“fd”, solid curve) and sign-preserving (“sp”, dashed curve) error models of the noisy adder.

The error probability of the noiseless decoder is also plotted (solid black curve): it can be seen that it increases rapidly from the initial value  $P_e^{(0)} = p_0$  and closely approaches the limit value  $P_e^{(\infty)} = 0.323$  after a few number of iterations. When the adder is noisy, the error probability increases during the first decoding iterations, and behaves similarly as in the noiseless case. It may approach the limit value from the noiseless case, but starts decreasing after some number of decoding iterations. However, it remains

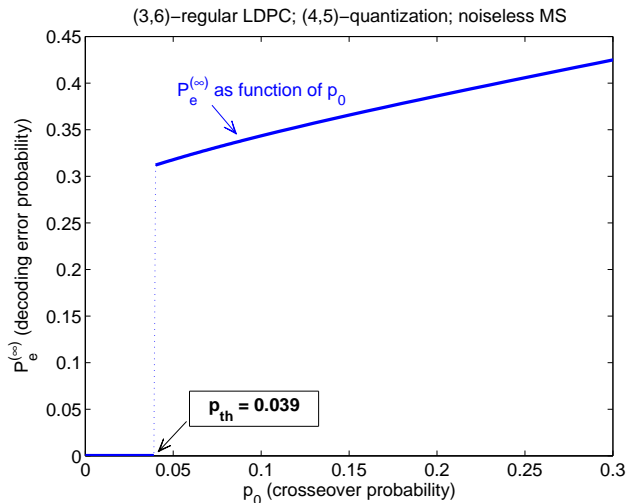


Figure 1. Asymptotic error probability  $P_e^{(\infty)}$  of the noiseless MS decoder as a function of  $p_0$

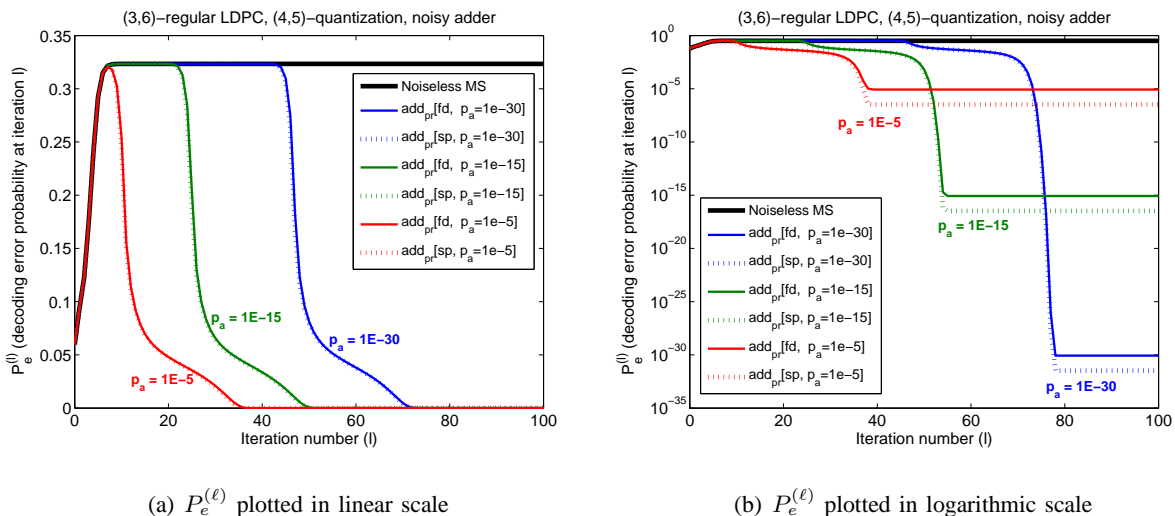


Figure 2. Effect of the noisy adder on the asymptotic performance of the MS decoder ( $p_0 = 0.06$ )

bounded above zero, according to the lower bounds from Proposition 1. This can be seen in Figure 2(b), where  $P_e^{(\ell)}$  plotted in logarithmic scale. The asymptotic values  $P_e^{(\infty)}$  and the corresponding lower-bounds values from Proposition 1 are shown in Table II. It can be seen that these bounds are tight, especially in the sign-preserving case.

The above behavior of the MS decoder is explained by the fact that the noise present in the adder helps the MS decoder to escape from fixed points attractors. Figure 3 illustrates the evolution of the probability mass function  $\tilde{C}^{(\ell)}$  for the noiseless decoder. At iteration  $\ell = 0$ ,  $\tilde{C}^{(0)}$  is supported in  $\pm 1$ , with  $\tilde{C}^{(0)}(-1) = p_0$  and  $\tilde{C}^{(0)}(+1) = 1 - p_0$ . It evolves during the iterative decoding, and reaches a fixed point of the density evolution for  $\ell = 20$ . Note that since all variable-nodes are of degree  $d_v = 3$ , it can



Table II  
ASYMPTOTIC ERROR PROBABILITY OF THE MS DECODING WITH NOISY ADDER ( $p_0 = 0.06$ )

		$p_a$	$10^{-30}$	$10^{-15}$	$10^{-5}$
full depth	$P_e^{(\infty)}$		$8.500 \times 10^{-31}$	$8.500 \times 10^{-16}$	$8.507 \times 10^{-6}$
	lower-bound		$5.167 \times 10^{-31}$	$5.167 \times 10^{-16}$	$5.167 \times 10^{-6}$
sign preserving	$P_e^{(\infty)}$		$3.333 \times 10^{-32}$	$3.333 \times 10^{-17}$	$3.333 \times 10^{-7}$
	lower-bound		$3.333 \times 10^{-32}$	$3.333 \times 10^{-17}$	$3.333 \times 10^{-7}$

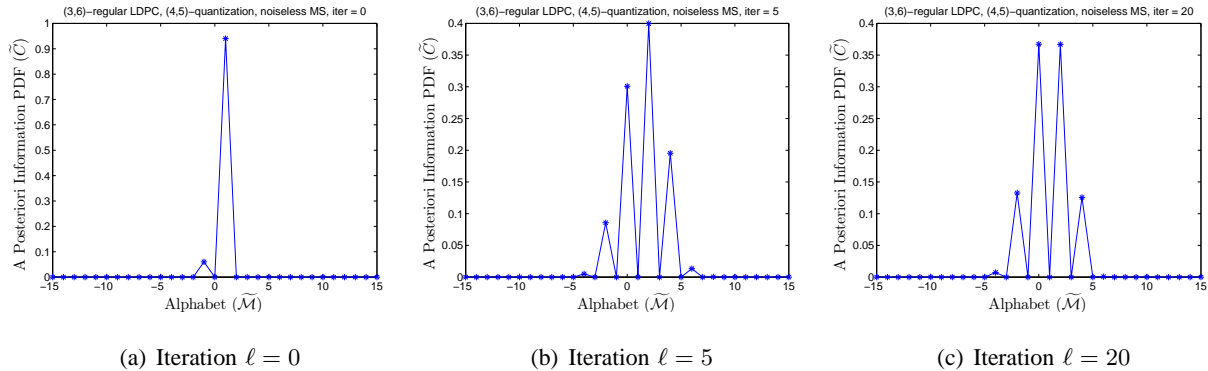


Figure 3. Probability mass function of the a posteriori information  $\tilde{C}^{(\ell)}$  (noiseless MS decoder)

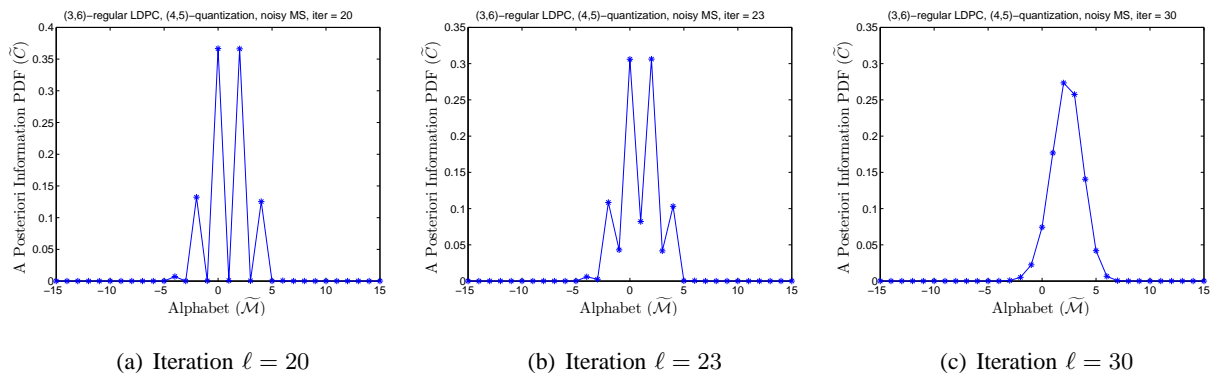


Figure 4. Probability mass function of the a posteriori information  $\tilde{C}^{(\ell)}$  (MS decoder with full-depth noisy adder,  $p_a = 10^{-15}$ )  
be easily seen that, for  $\ell \geq 1$ ,  $\tilde{C}^{(\ell)}$  is supported only on even values. These “gaps” in the probability mass function seem lead to favorable conditions for the occurrence of density-evolution fixed-points.

Figure 4 illustrates the evolution of the probability mass function  $\tilde{C}^{(\ell)}$  when the full-depth noisy adder with  $p_a = 10^{-15}$  is used within the MS decoder. At iteration  $\ell = 20$ ,  $\tilde{C}^{(\ell)}$  is virtually the same as in the noiseless case. However, the noisy adder allows the decoder to escape from this fixed-point, as it can be seen for iterations  $\ell = 23$  and  $\ell = 30$ . For  $\ell > 30$ , the  $\tilde{C}^{(\ell)}$  moves further on the right, until the corresponding error probability  $P_e^{(\ell)}$  reaches the limit value  $P_e^{(\infty)} = 8.5 \times 10^{-16}$ .

It is worth noting that neither the noisy comparator nor the XOR-operator can help the decoder to escape from fixed-point distributions, as they do not allow “filling the gaps” in the support of  $\tilde{C}^{(\ell)}$ .

We focus now on the useful region of the noisy MS decoder. We assume that only the adder is noisy, while the comparator and the XOR-operator are noiseless.

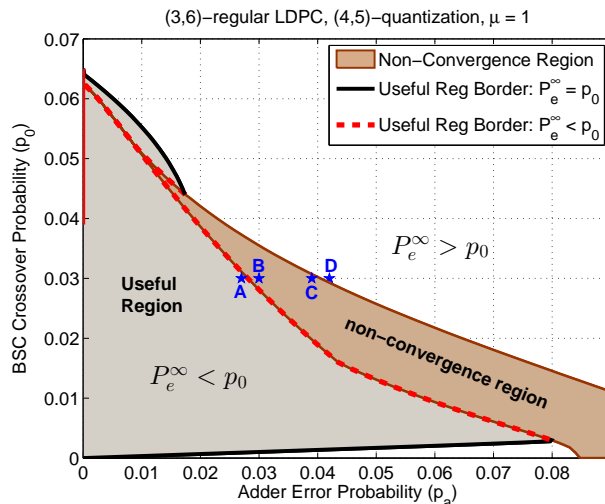


Figure 5. Useful and non-convergence regions of the MS decoder with sign-preserving noisy adder

The useful region for the sign-preserving noisy adder model is shown in Figure 5. The useful region is shaded in gray and delimited by either a solid black curve or a dashed red curve. Although one would expect that  $P_e^{(\infty)} = p_0$  on the border of the useful region, this equality only holds on the solid black border. On the dashed red border, one has  $P_e^{(\infty)} < p_0$ . The reason why the useful region does not extend beyond the dashed red border is that for points located on the other side of this border the sequence  $(P_e^{(\ell)})_{\ell > 0}$  is periodic, and hence it does not converge! The region shaded in brown in Figure 5 is the *non-convergence region* of the decoder. Note that the non-convergence region gradually narrows in the upper part, and there is a small portion of the useful region delimited by the non-convergence region on the left and the black border on the right. Finally, we note that points with  $p_a = 0$  (noiseless decoder) and  $p_0 > 0.039$  (threshold of the noiseless decoder) – represented by the solid red line superimposed on the vertical axis in Figure 5 – are excluded from the useful region. Indeed, for such points  $P_e^{(\infty)} > p_0$ ; however, for  $p_a$  greater than but close to zero, we have  $P_e^{(\infty)} \approx \frac{p_a}{2Q}$  (see Figure 2 and related discussion).

We exemplify the decoder behavior on four points located on one side and the other of the left and right boundaries of the non-convergence region. These points are indicated in Figure 5 by  $A$ ,  $B$ ,  $C$ , and  $D$ . For all the four points  $p_0 = 0.03$ , while  $p_a = 0.027, 0.03, 0.039$ , and  $0.042$ , respectively. The error

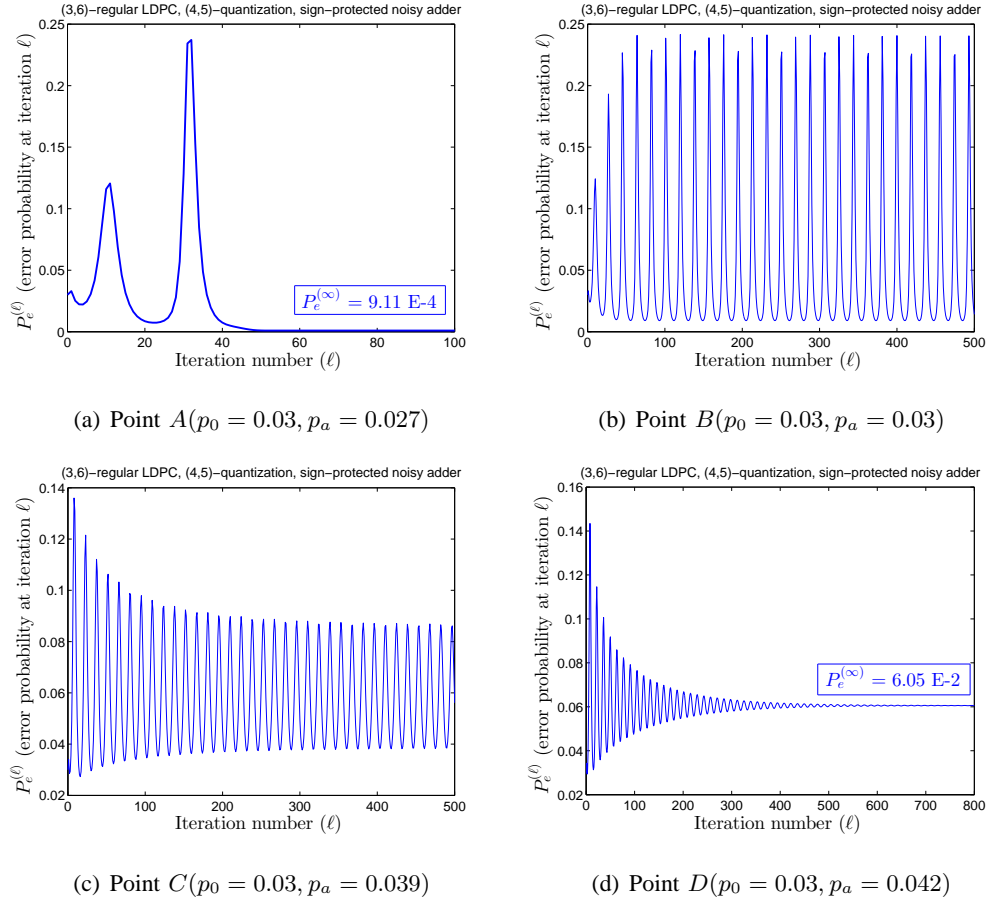


Figure 6. Decoding error probability  $P_e^{(\ell)}$  of the noisy MS decoder, for  $p_0 = 0.03$  and sign-preserving noisy adder with various  $p_a$  values

probability  $(P_e^{(\ell)})_{\ell>0}$  is plotted for each one of these points in Figure 6. The point  $A$  belongs to the useful region, and it can be seen from Figure 6(a) that  $(P_e^{(\ell)})_{\ell>0}$  converges to  $P_e^{(\infty)} = 9.11 \times 10^{-4} < p_0$ . For the point  $B$ , located just on the other side of the dashed red border of the useful region,  $(P_e^{(\ell)})_{\ell>0}$  exhibits a periodic behavior (although we only plotted the first 500 iterations, we verified the periodic behavior on the first  $5 \times 10^4$  iterations). Crossing the non-convergence region from left to the right, the amplitude between the inferior and superior limits of  $(P_e^{(\ell)})_{\ell>0}$  decreases (point C), until it reaches again a convergent behavior (point D). Note that  $D$  is outside the useful region, as  $(P_e^{(\ell)})_{\ell>0}$  converges to  $P_e^{(\infty)} = 0.0605 > p_0$ .

The non-convergence region gradually narrows in the upper part, and for  $0 \leq p_a < 0.01$  it takes the form of a *discontinuity line*:  $P_e^{(\infty)}$  takes values close to  $10^{-4}$  just below this line, and values greater than 0.05 above this line.

Note that points  $(p_a, p_0)$  with  $p_0 < \frac{p_a}{2Q} = \frac{p_a}{30}$  cannot belong to the useful region, since from Proposi-

tion 1 we have  $P_e^{(\infty)} \geq \frac{p_a}{2Q} > p_0$ . Moreover, we note that the bottom border of the useful region (solid black curve) is virtually identical to, but slightly above, the line defined by  $p_0 = \frac{p_a}{2Q}$ .

2) *Optimization of the quantization map*: In this section we show that the decoder performance can be significantly improved by using an appropriate choice of the channel scale factor  $\mu$ . Figure 7 shows the threshold values for the noiseless and several noisy decoders with channel scale factors  $\mu \in \{1, 2, \dots, 7\}$ . For the noisy decoders, the threshold values are computed for a target error probability  $\eta = 10^{-5}$  (see Equation (19)).

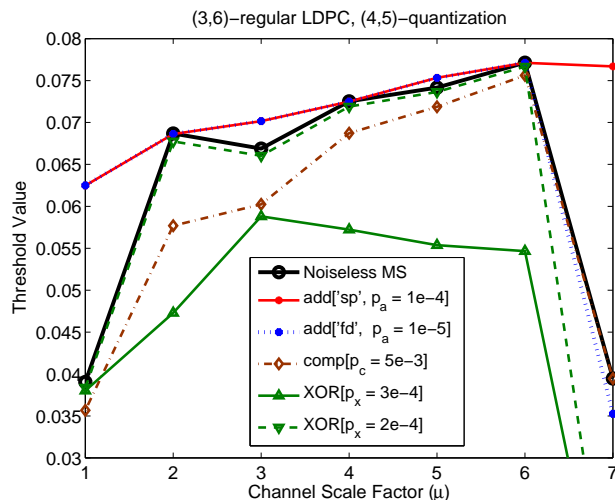


Figure 7. Threshold values of noiseless and noisy MS decoders with various channel scale factors (for noisy decoders, threshold values correspond to a target error probability  $\eta = 10^{-5}$ )

The solid black curve in Figure 7 correspond to the noiseless decoder. The solid red curve and the dotted blue curve correspond to the MS decoder with sign-preserving noisy adder and full-depth noisy adder, respectively. The adder error probability is  $p_a = 10^{-4}$  for the sign-preserving noisy adder, and  $p_a = 10^{-5}$  for the full-depth adder<sup>8</sup>. The two curves are superimposed for  $1 \leq \mu \leq 6$ , and differ only for  $\mu = 7$ . The corresponding threshold values are equal to those obtained in the noiseless case for  $\mu \in \{2, 4, 6\}$ . For  $\mu \in \{1, 3, 5\}$ , the MS decoders with noisy-adders exhibit better thresholds than the noiseless decoder. This is due to the fact that the messages alphabet  $\mathcal{M}$  is underused by the noiseless decoder, since all the exchanged messages are necessarily odd (recall that all variable-nodes are of degree

<sup>8</sup>Note that according to Proposition 1, a necessary condition to achieve a target error probability  $P_e^{(\infty)} \leq \eta = 10^{-5}$  is  $p_a \leq 2\tilde{Q}\eta = 3 \times 10^{-4}$  for the signed-preserving adder, and  $p_a \leq 2\eta \frac{2\tilde{Q}+1}{2Q} = 2.07 \times 10^{-5}$  for the full-depth adder.

$d_v = 3$ ). For the MS decoders with noisy adders, the noise present in the adders leads to a more efficient use of the messages alphabet, which allows the decoder to escape from fixed-point attractors and hence results in better thresholds (Section VI-A1).

Figure 7 also shows a curve corresponding to the MS decoder with a noisy comparator having  $p_c = 0.005$ , and two curves for the MS decoder with noisy XOR-operators, having respectively  $p_x = 2 \times 10^{-4}$  and  $p_x = 3 \times 10^{-4}$ .

Concerning the noisy XOR-operator, it can be seen that the threshold values corresponding to  $p_x = 2 \times 10^{-4}$  are very close to those obtained in the noiseless case, except for  $\mu = 7$  (the same holds for values  $p_x < 2 \times 10^{-4}$ ). However, a significant degradation of the threshold can be observed when slightly increasing the XOR error probability to  $p_x = 3 \times 10^{-4}$ . Moreover, although not shown in the figure, it is worth mentioning that for  $p_x \geq 5 \times 10^{-4}$ , the target error probability  $\eta = 10^{-5}$  can no longer be reached (thus, all threshold values are equal to zero).

Finally, we note that except for the noisy XOR-operator with  $p_x = 3 \times 10^{-4}$ , the best choice of the channel scale factor is  $\mu = 6$ . For the noisy XOR-operator with  $p_x = 3 \times 10^{-4}$ , the best choice of the channel scale factor is  $\mu = 3$ . This is rather surprising, as in this case the messages alphabet is underused by the decoder: all the exchanged messages are odd, and the fact that the XOR-operator is noisy does not change their parity.

**Assumption:** In the following sections, we will investigate the impact of the noisy adder, comparator and XOR-operator on the MS decoder performance, assuming that the channel scale factor is  $\mu = 6$ .

3) *Study of the impact of the noisy adder (quantization map  $\mathbf{q}_6$ ):* In order to evaluate the impact of the noisy adder on the MS decoder performance, the useful region and the  $\eta$ -threshold regions have been computed, assuming that only the adders within the VN-processing step are noisy ( $p_a > 0$ ), while the CN-processing step is noiseless ( $p_x = p_c = 0$ ). This regions are represented in Figure 8, for both sign-preserving and full-depth noisy adder models.

The useful region is delimited by the solid black curve. The vertical lines delimit the  $\eta$ -threshold regions, for  $\eta = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$  (from right to the left).

Note that unlike the case  $\mu = 1$  (Section VI-A1), there is no non-convergence region when the channel scale factor is set to  $\mu = 6$ . Hence, the border of the useful region corresponds to points  $(p_a, p_0)$  for which  $P_e^{(\infty)} = p_0$ . However, it can be observed that there is still a *discontinuity line* (dashed red curve) inside the useful region. This discontinuity line does not hide a periodic (non-convergent) behavior, but it is due to the occurrence of an *early plateau phenomenon* in the convergence of  $(P_e^{(\ell)})_\ell$ . This phenomenon is

illustrated in Figure 9, where the error probability  $(P_e^{(\ell)})_\ell$  is plotted as a function of the iteration number  $\ell$ , for the two points A and B from Figure 8(a). For point A, it can be observed that the error probability  $P_e^{(\ell)}$  reaches a first plateau for  $\ell \approx 50$ , then drops to  $3.33 \times 10^{-6}$  for  $\ell \geq 250$ . For point B,  $P_e^{(\ell)}$  behaves in a similar manner during the first iterations, but it does not decrease below the plateau value as  $\ell$  goes to infinity. Although we have no analytic proof of this fact, it was numerically verified for  $\ell \leq 5 \times 10^5$ .

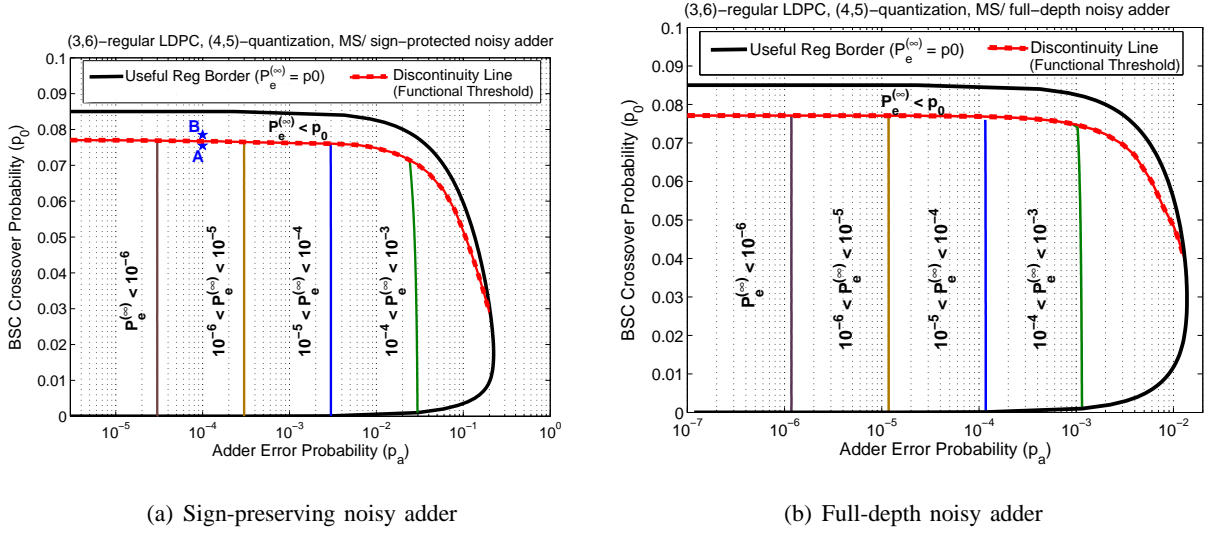


Figure 8. Useful and  $\eta$ -threshold regions of the MS decoder with noisy adder

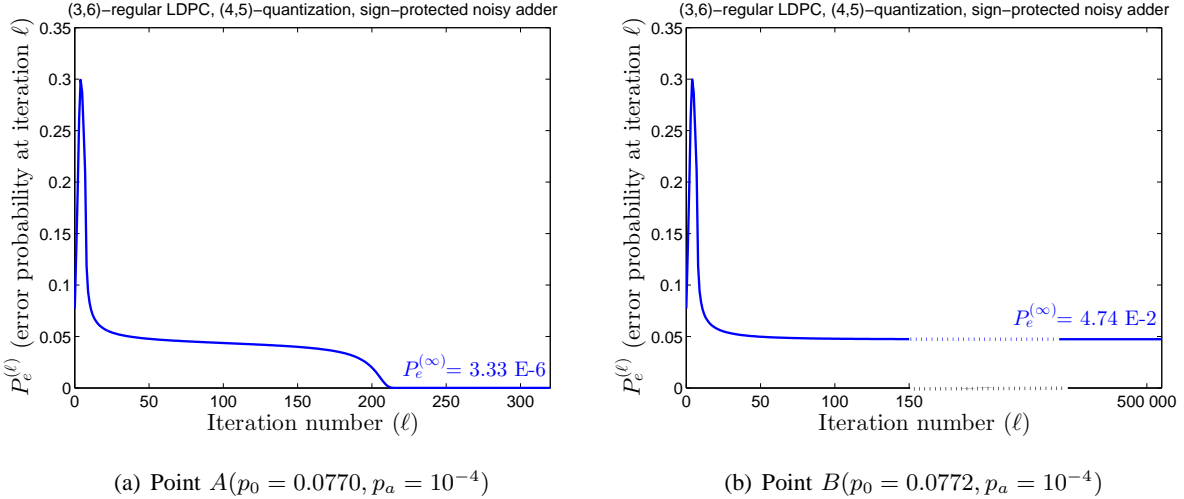


Figure 9. Illustration of the early plateau phenomenon (points A and B from Figure 8(a))

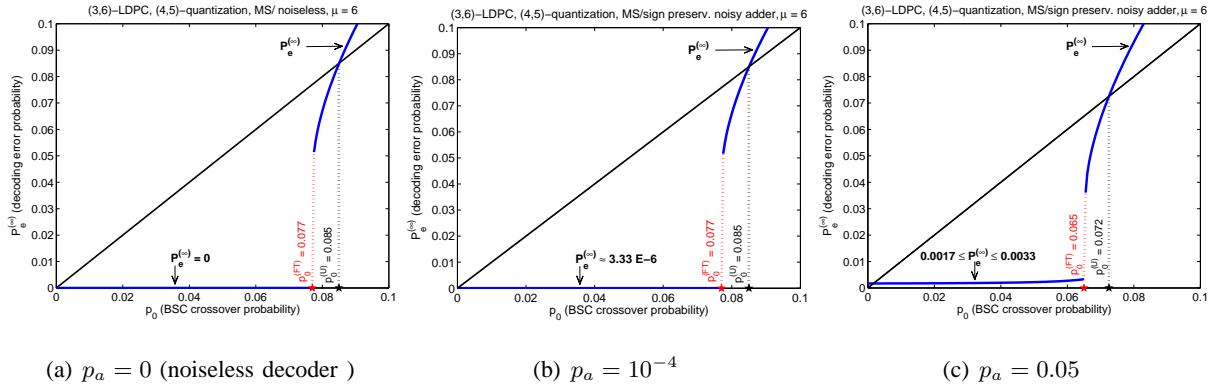


Figure 10. Asymptotic error probability  $P_e^{(\infty)}$  as a function of  $p_0$ ; noiseless and noisy MS decoder with sign-preserving noisy adder

In Figure 10, we plotted the asymptotic error probability  $P_e^{(\infty)}$  as a function of  $p_0$ , for the noiseless decoder ( $p_a = 0$ ), and for the sign-preserving noisy adder with error probability values  $p_a = 10^{-4}$  and  $p_a = 0.05$ . In each plot we have also represented two points  $p_0^{(U)}$  and  $p_0^{(FT)}$ , corresponding respectively to the values of  $p_0$  on the upper-border of the useful region, and on the discontinuity line. Hence,  $p_0^{(FT)}$  coincides with the classical threshold of the MS decoder in the noiseless case, and it is equal to the functional threshold defined in Section V-D in case of noisy decoders. In the following, the sub-region of the useful region located below the discontinuity line will be referred to as the *functional region*. Within this region, if the adder error probability is small enough, it can be observed that:

- (a) For the sign-preserving adder:  $P_e^{(\infty)} \approx \frac{p_a}{30}$ , for  $p_a \lesssim 3 \times 10^{-2}$ , which corresponds to the value given by the lower-bound ( $\frac{1}{2Q}p_a = \frac{1}{30}p_a$ ) from Proposition 1.
- (b) For the full-depth adder:  $P_e^{(\infty)} \approx 1.17p_a$ , for  $p_a \lesssim 10^{-3}$ , which is about twice higher than the value given by the lower-bound ( $\frac{1}{2}p_a + \frac{1}{4Q}p_a = 0.52p_a$ ) from Proposition 1.

Finally, we note that by protecting the sign of the noisy adder, the useful region is expanded by a factor of roughly  $2\tilde{Q}$ , representing an exponential improvement with respect to the number of bits of the adder (see also the discussion following the proof of Proposition 1).

4) *Study of the impact of the noisy XOR-operator (quantization map  $\mathbf{q}_6$ ):* The useful region and the  $\eta$ -threshold regions of the decoder, assuming that only the XOR-operator used within the CN-processing step is noisy, are plotted in Fig. 11. Similar to the noisy-adder case, a discontinuity (functional threshold) line can be observed inside the useful region, which delimits the *functional region* of the decoder.

Comparing the  $\eta$ -threshold regions from Figure 8 and Figure 11, it can be observed that in order to achieve a target error probability  $P_e^{(\infty)} \leq 10^{-6}$ , the error probability parameters of the noisy adder and of the noisy XOR-operator must satisfy:

- $p_a < 1.17 \times 10^{-6}$ , for the full-depth noisy-adder;
- $p_a < 3 \times 10^{-5}$ , for the sign-preserving noisy-adder;
- $p_x < 7 \times 10^{-5}$ , for the noisy XOR-operator.

(moreover, values of  $p_x$  up to  $1.4 \times 10^{-4}$  are tolerable if  $p_0$  is sufficiently small)

The most stringent requirement concerns the error probability of the full-depth noisy-adder, thus we may consider that it has the most negative impact on the decoder performance. On the other hand, the less stringent requirement concerns the error probability of the noisy XOR-operator.

Finally, it is worth noting that in practical cases the value of  $p_x$  should be significantly lower than the value of  $p_a$  (given the high number of elementary gates contained in the adder). Moreover, since the XOR-operators used to compute the signs of CN messages represent only a small part of the decoder, this part of the circuit could be made reliable by using classical fault-tolerant methods, with a limited impact on the overall decoder design.

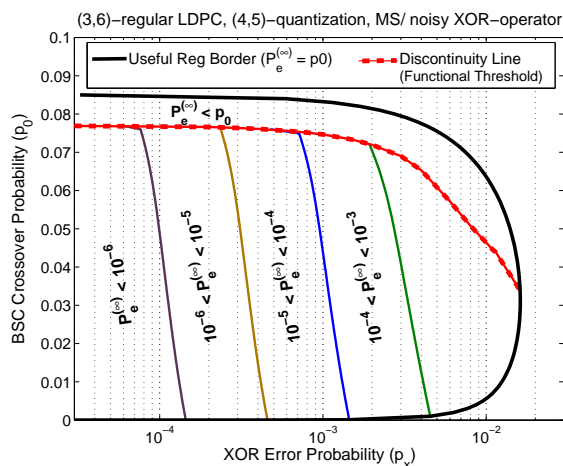


Figure 11. Useful and  $\eta$ -threshold regions of the MS decoder with noisy XOR-operator

5) *Study of the impact of the noisy comparator (quantization map  $\mathbf{q}_6$ ):* This section investigates the case when comparators used within the CN-processing step are noisy ( $p_c > 0$ ), but  $p_a = p_x = 0$ . Contrary to the previous cases, this case exhibits a “classical” threshold phenomenon, similar to the noiseless case: for a given  $p_c > 0$ , there exists a  $p_0$ -threshold value, denoted by  $p_0^{(\text{TH})}$ , such that  $P_e^{(\infty)} = 0$  for any  $p_0 < p_0^{(\text{TH})}$ .

The threshold value  $p_0^{(\text{TH})}$  is plotted as a function of  $p_c$  in Figure 12. The functional region of the decoder is located below the threshold curve, and  $P_e^{(\infty)} = 0$  for any point within this region. In particular, it can



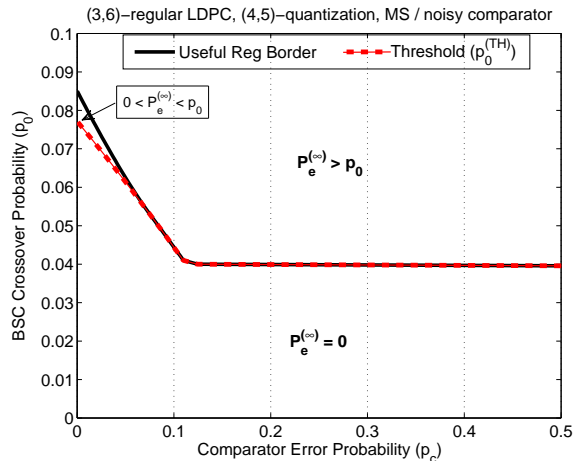


Figure 12. Useful region and threshold curve of the MS decoder with noisy comparator

be seen that  $P_e^{(\infty)} = 0$  for any  $p_0 \lesssim 0.039$  and any  $p_c > 0$ . Although such a threshold phenomenon might seem surprising for a noisy decoder, it can be easily explained. The idea behind is that in this case the crossover probability of the channel is small enough, so that in the CN-processing step only the sign of check-to-variable messages is important, but not their amplitudes. In other words a decoder that only computes (reliably) the signs of check-node messages and randomly chooses their amplitudes, would be able to perfectly decode the received word.

Finally, we note that the useful region of the decoder extends slightly above the threshold curve: for  $p_c$  close to 0, there exists a small region above the threshold curve, within which  $0 < P_e^{(\infty)} < p_0$ .

### B. Numerical results for the BI-AWGN channel

For the BI-AWGN, the channel output is given by  $y = x + z$ , where  $x \in \{\pm 1\}$  is the channel input and  $z$  is the additive white Gaussian noise with variance  $\sigma^2$ . Threshold values and useful regions of the decoder will be described in terms of Signal to Noise Ratio (SNR), defined by  $\text{SNR} = -10 \log_{10}(\sigma^2)$ .

For a given channel scale factor  $\mu$ , the quantization map  $\mathbf{q}_\mu$  is defined by  $\mathbf{q}_\mu(y) = \mathbf{s}_M([\mu \cdot y])$ , where  $[\mu \cdot y]$  denotes the nearest integer to  $\mu \cdot y$ , and  $\mathbf{s}_M$  is the saturation map (see also Equation (12)).

Similar to the BSC case, the choice of the channel scale factor  $\mu$  may significantly impact the decoder performance. Hence, we start first by optimizing the channel scale factor value, and then we investigate the impact of the different noisy components on the decoder performance.

**Remark:** For the BI-AWGN channel we denote by  $p_0 \stackrel{\text{def}}{=} P_e^{(0)}$  the error probability at iteration 0, which is, by definition, the probability of the *a priori* information  $\gamma = \mathbf{q}_\mu(y)$  being in error. Hence,

$p_0 = \sum_{z=-Q}^{-1} C(z) + \frac{1}{2}C(0)$ . Using Equation (14) it follows that:

$$p_0 = 1 - \frac{1}{2} \left[ q \left( \frac{-0.5 - \mu}{\mu\sigma} \right) + q \left( \frac{0.5 - \mu}{\mu\sigma} \right) \right] \quad (23)$$

1) *Optimization of the quantization map*: The goal of this section is to provide an optimal choice of the channel scale factor  $\mu$ . Figure 13 shows the threshold SNR values for the noiseless and several noisy decoders for channel scale factors  $\mu$  varying within the interval  $[1, 7]$ . For the noisy decoders, the threshold values are computed for a target error probability  $\eta = 10^{-5}$  (see Equation (19)).

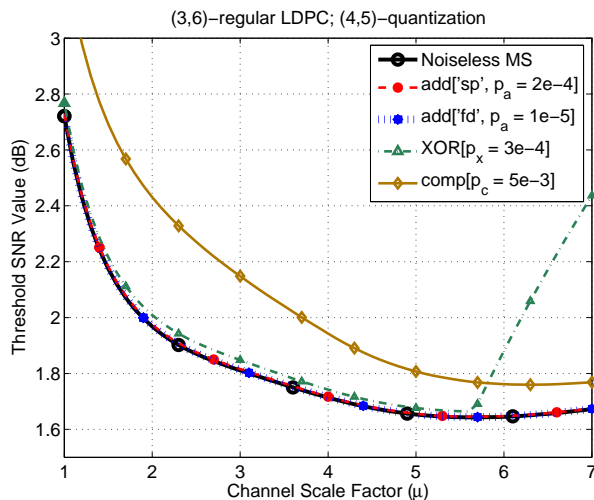


Figure 13. Threshold SNR values of noiseless and noisy decoders with various channel scale factors (for noisy decoders, threshold values correspond to a target error probability  $\eta = 10^{-5}$ )

The solid black curve in Figure 13 correspond to the noiseless decoder. The dashed red curve and the dotted blue curve correspond to the MS decoder with sign-preserving noisy adder and full-depth noisy adder, respectively. The adder error probability is  $p_a = 2 \times 10^{-4}$  for the sign-preserving noisy adder, and  $p_a = 10^{-5}$  for the full-depth adder<sup>9</sup>. These three curves are virtually indistinguishable.

Figure 13 also shows two curves corresponding respectively to the MS decoder with a noisy XOR-operator ( $p_x = 2 \times 10^{-4}$ ) and to the MS decoder with a noisy comparator ( $p_c = 0.005$ ). Finally, we note that in all cases the best choice of the channel scale factor is  $\mu \approx 5.5$ .

**Assumption:** In the following sections, we will investigate the impact of the noisy adder, comparator and XOR-operator on the MS decoder performance, assuming that the channel scale factor is  $\mu = 5.5$ .

<sup>9</sup>Note that according to Proposition 1, a necessary condition to achieve a target error probability  $P_e^{(\infty)} \leq \eta = 10^{-5}$  is  $p_a \leq 2\tilde{Q}\eta = 3 \times 10^{-4}$  for the signed-preserving adder, and  $p_a \leq 2\eta \frac{2\tilde{Q}+1}{2\tilde{Q}} = 2.07 \times 10^{-5}$  for the full-depth adder.

2) *Study of the impact of the noisy adder:* Useful and  $\eta$ -regions of the MS decoder with noisy adders are represented in Figure 14, for both sign-preserving and full-depth noisy adder models. The useful region is delimited by the solid black curve, while vertical lines delimit the  $\eta$ -threshold regions, for  $\eta = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$  (from right to the left). The *functional threshold* of the decoder is also displayed by a red dashed curve.

Figure 15 shows the input and output error probabilities of the decoder ( $p_0$  and  $P_e^{(\infty)}$ ) as functions of the SNR value, for the sign-preserving and full-depth noisy adder models with  $p_a = 10^{-4}$ . The two intersection points between the two curves correspond to the points on the lower and upper borders of the

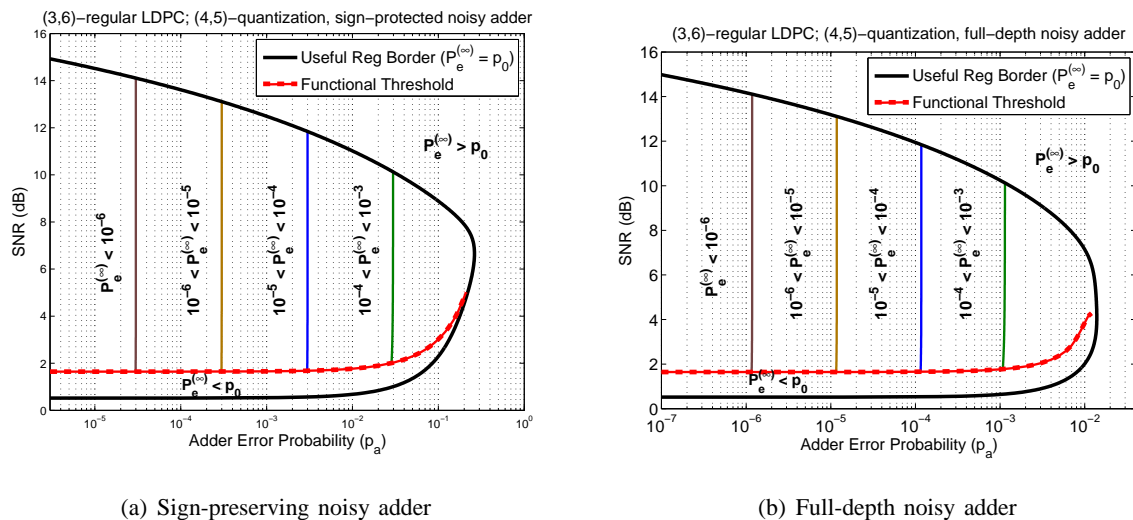


Figure 14. Useful and  $\eta$ -threshold regions of the MS decoder with noisy adder (BI-AWGN)

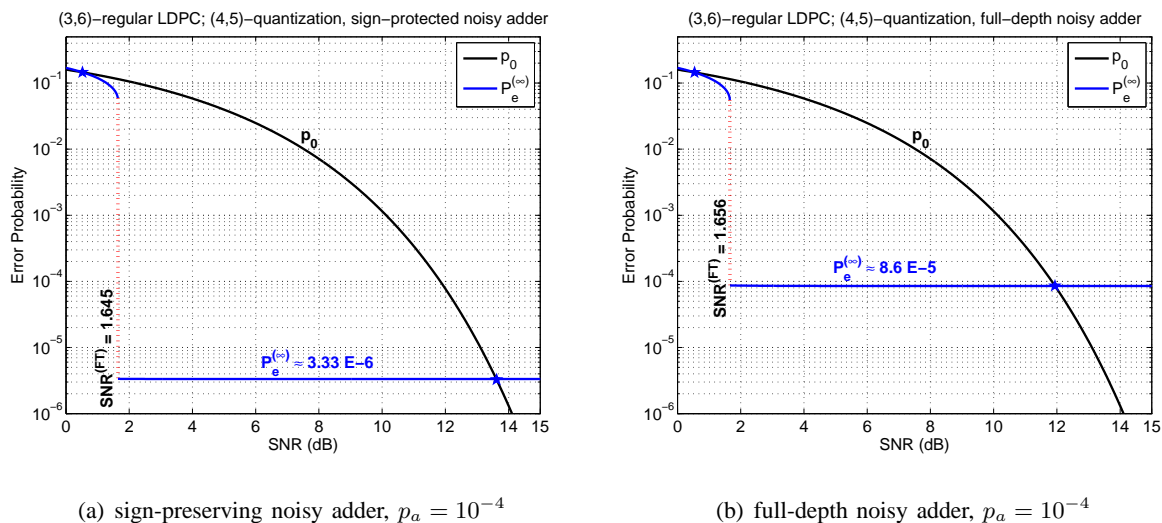


Figure 15. Asymptotic error probability  $P_e^{(\infty)}$  of the MS decoder with noisy-adder as a function of the SNR

the useful region in Figure 14, for  $p_a = 10^{-4}$ . The discontinuity point of the  $P_e^{(\infty)}$  curve corresponds to the functional threshold value in Figure 14, for  $p_a = 10^{-4}$ .

3) *Study of the impact of the noisy XOR-operator and noisy comparator:* The useful region and the  $\eta$ -threshold regions of the MS decoder, assuming that only the XOR-operator used within the CN-processing step is noisy, are plotted in Fig. 16. The *functional threshold* of the decoder is also displayed by a red dashed curve.

The case of a noisy comparator is illustrated in Figure 17. Similar to the BSC channel, this case exhibits a “classical” threshold phenomenon: for any SNR value above the functional threshold curve, one has  $P_e^{(\infty)} = 0$ .

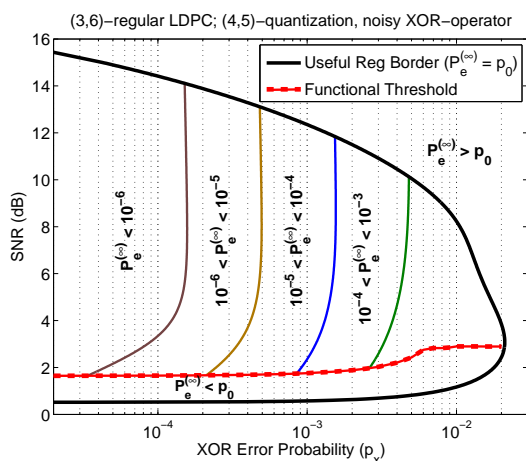


Figure 16. Useful and  $\eta$ -threshold regions of the MS decoder with noisy XOR-operator (BI-AWGN)

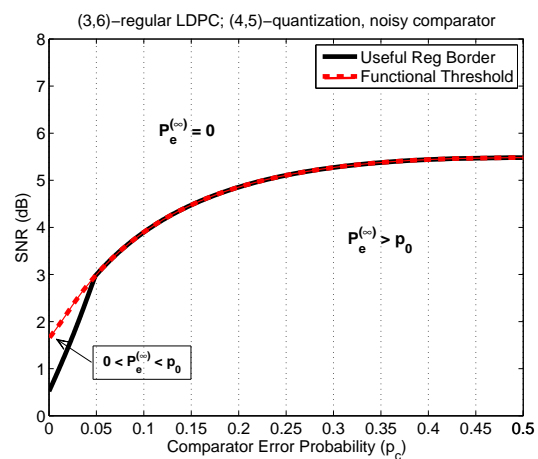


Figure 17. Useful region and threshold curve of the MS decoder with noisy comparator (BI-AWGN)

## VII. FINITE LENGTH PERFORMANCE OF MIN-SUM BASED DECODERS

The goal of this section is twofold:

- (1) To corroborate the asymptotic analysis through finite-length simulations;
- (2) To investigate ways of increasing the robustness of the MS decoder to hardware noise.

**Assumption:** Unless otherwise stated, the (3,6)-regular LDPC code with length  $N = 1008$  bits from [25] will be used for finite length simulations throughout this section.

### A. Practical implementation and early stopping criterion

First of all, we note that the practical implementation of the noisy MS decoder differs slightly from the one presented in Algorithm 2:

- The order of the **VN-processing** and **AP-update** steps is inverted;
- The variable-to-check node messages are computed by subtracting the incoming check-to-variable message from the corresponding a posteriori information value:

```

for all  $n = 1, \dots, N$  do ▷ AP-update
     $\tilde{\gamma}_n = \mathbf{a}_{\text{pr}}(\{\gamma_n\} \cup \{\beta_{m,n}\}_{m \in \mathcal{H}(n)});$ 

for all  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  do ▷ VN-processing
     $\alpha_{m,n} = \mathbf{a}_{\text{pr}}(\tilde{\gamma}_n, -\beta_{m,n});$ 
     $\alpha_{m,n} = \mathbf{s}_{\mathcal{M}}(\alpha_{m,n});$ 

```

For floating-point noiseless decoders, the two ways of computing the variable-to-check messages are completely equivalent. However, this equivalence does not hold anymore for finite-precision (noisy or noiseless) decoders, because of saturation effects and, in case of noisy decoders, of probabilistic computations. We note that the practical implementation might result in a degradation of the decoder performance compared to the “Density-Evolution like” implementation (Algorithm 2), since each variable-to-check node message *encompasses*  $d_v + 1$  additions ( $d_v$  additions to compute  $\tilde{\gamma}_n$  and one subtraction).

Finally, it is worth noting that the density-evolution analysis cannot be applied to the practical implementation, due to the fact that in the VN-processing step, the computation of variable-to-check messages  $\alpha_{m,n} = \mathbf{a}_{\text{pr}}(\{\gamma_n\}, -\beta_{m,n})$  involves two correlated variables, namely  $\gamma_n$  and  $\beta_{m,n}$ .

1) *Early stopping criterion (syndrome check)*: As described in Algorithm 2, each decoding iteration also comprises a *hard decision* step, in which each transmitted bit is estimated according to the sign of the a posteriori information, and a *syndrome check* step, in which the syndrome of the estimated word is computed.

Both steps are assumed to be *noiseless*, and the syndrome check step acts as an *early stopping criterion*: the decoder stops when whether the syndrome is  $+1$  (the estimated word is a codeword) or a maximum number of iterations is reached. We note however that the syndrome check step is optional and, if missing, the decoder stops when the maximum number of iterations is reached.

**Remark:** The reason why we stress the difference between the MS decoder with and without the syndrome check step is because, as we will see shortly, the *noiseless* early stopping criterion may significantly improve the bit error rate performance of the *noisy* decoder in the error floor region.

**Assumptions:**

- Unless otherwise stated, the MS decoder is assumed to implement the *noiseless* stopping criterion (syndrome check step).
- The maximum number of decoding iterations is fixed to 100 throughout this section.

*B. Corroboration of the asymptotic analysis through finite-length simulations*

We start by analyzing the finite-length decoder performance over the BSC channel. Figure 18 shows the bit error rate (BER) performance of the finite-precision MS decoder (both noiseless and noisy) with various channel scale factors. For comparison purposes, we also included the BER performance of the Belief-Propagation decoder (solid black curve, no markers) and of the infinite-precision MS decoder (dashed blue curve, no markers).

It can be observed that the worst performance is achieved by the infinite-precision MS decoder (!) and the finite-precision noiseless MS decoder with channel scale factor  $\mu = 1$  (both curves are virtually indistinguishable). The BER performance of the latter improves significantly when using a sign-preserving noisy adder with error probability  $p_a = 0.001$  (dashed red curve with empty circles).

For a channel scale factor  $\mu = 6$ , both noiseless and noisy decoders have almost the same performance (solid and dashed green curves, with triangular markers). Remarkably, the achieved BER is very close to the one achieved by the Belief-Propagation decoder!

These results corroborate the asymptotic analysis from Section VI-A concerning the channel scale factor optimization.

1) *Error floor performance:* Surprisingly, the BER curves of the noisy decoders from Figure 18 do not show any error floor down to  $10^{-7}$ . However, according to Proposition 1, the decoding error probability should be lower-bounded by  $P_e^{(\ell)} \geq \frac{1}{2Q} p_a = 3.33 \times 10^{-5}$  (see also the  $\eta$ -threshold regions in Figure 8(a)).

The fact that the observed decoding error probability may decrease below the above lower-bound is due to the early stopping criterion (syndrome check step) implemented within the MS decoder. Indeed,

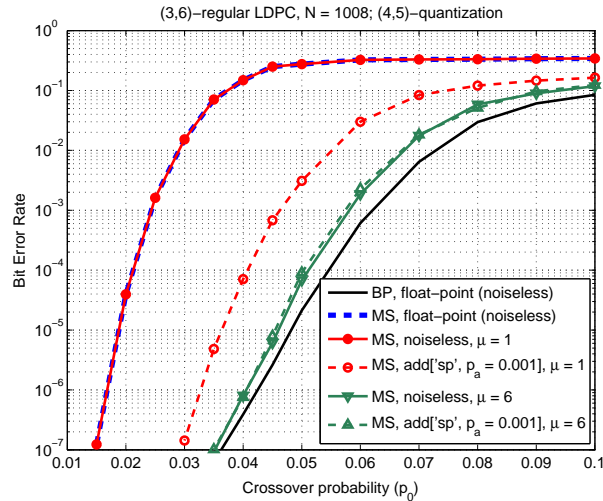


Figure 18. BER performance of noiseless and noisy MS decoders with various channel scale factors

as we observed in the previous section, the above lower-bound is tight, when  $\ell$  (the iteration number) is sufficiently large. Therefore, as the iteration number increases, the expected number of erroneous bits gets closer and closer to  $\frac{1}{2Q}p_a N = 0.034$ , and the probability of not having any erroneous bit within one iteration approaches  $\left(1 - \frac{1}{2Q}p_a\right)^N = 0.967$ . As the decoder performs more and more iterations, it will eventually reach an error free iteration. The absence of errors is at once detected by the noiseless syndrome check step, and the decoder stops.

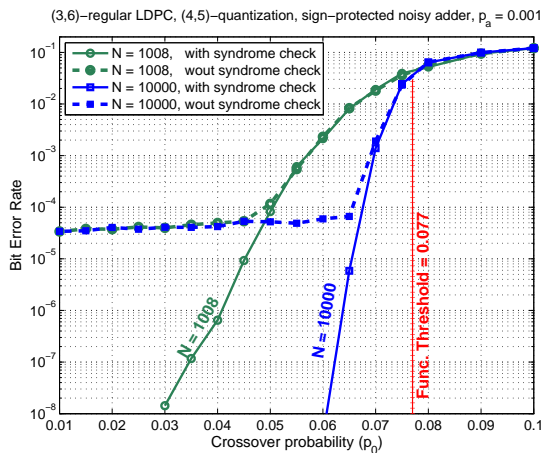


Figure 19. BER performance with and without early stopping criterion (MS decoder with sign-preserving noisy adder,  $p_a = 0.001$ )

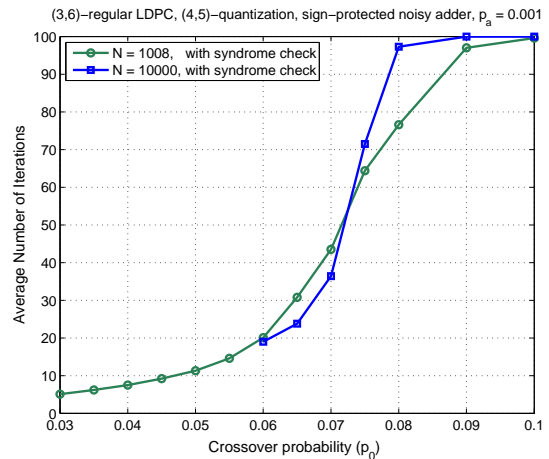


Figure 20. Average number of decoding iterations with early stopping criterion (MS decoder with sign-preserving noisy adder,  $p_a = 0.001$ )

To illustrate this behavior, we plotted the Figure 19 the BER performance of the noisy MS decoder, with and without early stopping criterion. The noisy MS decoder comprises a sign-preserving noisy adder with  $p_a = 0.001$ , while the comparator and the XOR-operator are assumed to be noiseless ( $p_c = p_x = 0$ ). Two codes are simulated, the first with length  $N = 1008$  bits, and the second with length  $N = 10000$  bits. In case that the noiseless early stopping criterion is implemented (solid curves), it can be seen that none of the BER curves show any error floor down to  $10^{-8}$ . However, if the early stopping criterion is not implemented (dashed curves), corresponding BER curves exhibit an error floor at  $\approx 3.33 \times 10^{-5}$ , as predicted by Proposition 1.

In Figure 20 we plotted the average number of decoding iterations in case that the early stopping criterion is implemented. It can be seen that the average number of decoding iterations decreases with the channel crossover probability  $p_0$ , or equivalently, with the achieved bit error rate. However, for a fixed BER – say BER =  $10^{-6}$ , achieved either at  $p_0 \approx 0.04$  for the code with  $N = 1008$ , or at  $p_0 \approx 0.063$  for the code with  $N = 10000$  – the average number of iterations is about 8 for the first code and about 21 for the second. Note that in case the early stopping criterion is not implemented, both codes have nearly the same performance for the above  $p_0$  values. Thus, when the early stopping criterion is implemented, the decoder needs to perform more iterations to eventually reach an error free iteration when  $N = 10000$ , which explains the increased average number of decoding iterations.

2) *Further results on the finite-length performance:* In this section we investigate the finite-length performance when all the MS components (adder, comparator, and XOR-operator) are noisy. In order to reduce the number of simulations, we assume that  $p_a = p_c \geq p_x$ . Concerning the noisy adder, we evaluate the BER performance for both the sign-preserving and the full-depth error models. Simulation results are presented in Figure 21. The error probability of the XOR-operator is  $p_x = 0.0001$  in sub-figures 21(a) and 21(b), and  $p_x = 0.001$  in sub-figures 21(c) and 21(d). The noisy adder is sign-preserving in sub-figures 21(a) and 21(c), and full-depth in sub-figures 21(b) and 21(d).

In case the noisy-adder is sign-preserving, it can be seen that the MS decoder can provide reliable error protection for all the noise parameters that have been simulated. Of course, depending on the error probability parameters of the noisy components, there is a more or less important degradation of the achieved BER with respect to the noiseless case. But in all cases the noisy decoder can achieve a BER less than  $10^{-7}$ . This is no longer true for the full-depth noisy adder: it can be seen that for  $p_c = p_a \geq 0.005$ , the noisy decoder cannot achieve bit error rates below  $10^{-2}$ .



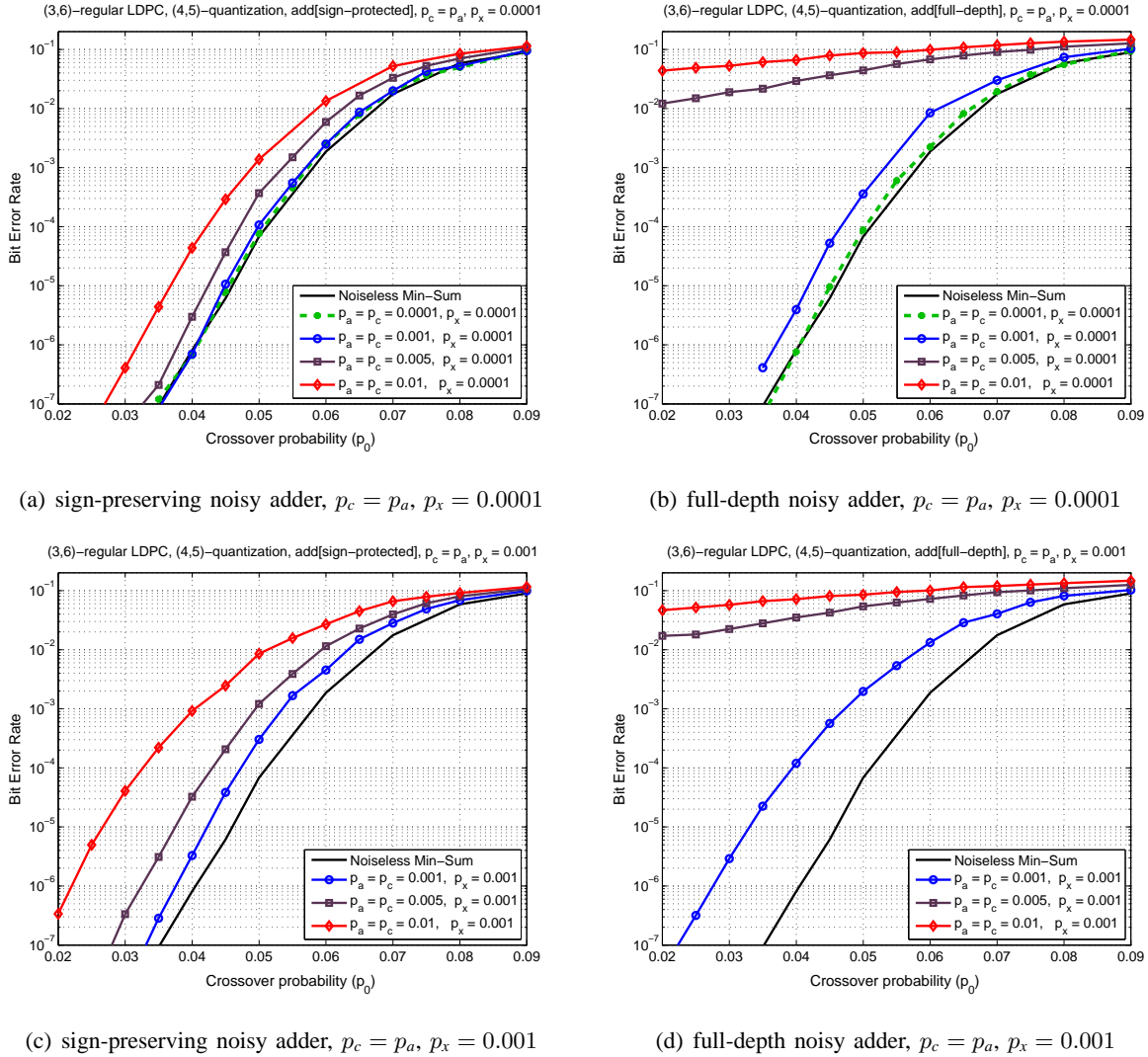


Figure 21. BER performance of the noisy MS decoder with various noise parameters

### C. Noisy Self-Corrected Min-Sum decoder

In this section we investigate the finite-length performance of the Self-Corrected Min-Sum (SCMS) decoder [26]. The objective is to determine if a correction circuit “plugged into” the noisy MS decoder can improve the robustness of the decoder to hardware noise.

The specificity of the SCMS decoder is to *erase* (i.e. set to zero) any variable-to-check message that changes its sign between two consecutive iterations. However, in order to avoid erasures propagation, a message cannot be erased if it has also been erased at the previous iteration. Hence, the SCMS decoder performs the same computations as the noisy MS, except that the **VN processing** step further includes

a *correction step*, as follows<sup>10</sup>:

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do**  $\triangleright$  VN-processing

$\alpha_{m,n}^{(\ell)} = \mathbf{s}_{\mathcal{M}} \left( \mathbf{a}_{\text{pr}} \left( \tilde{\gamma}_n^{(\ell)}, -\beta_{m,n}^{(\ell)} \right) \right);$

**if**  $\text{sgn} \left( \alpha_{m,n}^{(\ell)} \right) \neq \text{sgn} \left( \alpha_{m,n}^{(\ell-1)} \right)$  and  $\alpha_{m,n}^{(\ell-1)} \neq 0$

$\alpha_{m,n}^{(\ell)} = 0;$

**end**

The body enclosed between the **if** condition and the matching **end** is referred to as the *correction step*. In practical implementations, one needs to store the signs of the variable-to-check node messages and to keep a record of messages that have been erased by the self-correction step. We use the following notation:

- $s_{m,n}^{(\ell)} = \text{sgn} \left( \alpha_{m,n}^{(\ell)} \right)$ , the sign of the message  $\alpha_{m,n}^{(\ell)}$ ;
- $e_{m,n}^{(\ell)} \in \{0, 1\}$ , with  $e_{m,n}^{(\ell)} = 1$  if and only if the corresponding variable-to-check message has been erased at iteration  $\ell$ ; for  $\ell = 0$ , these values are all initialized as zero.
- $\text{SCU}(s_1, s_2, e) \stackrel{\text{def}}{=} (s_1 \oplus s_2) \otimes (1 \oplus e)$ , for any  $s_1, s_2, e \in \{0, 1\}$ , where  $\oplus$  denotes the XOR operation (sum modulo 2) and  $\otimes$  denotes the AND operation (product). Clearly  $\text{SCU}(s_1, s_2, e) = 1$  if and only if  $s_1 \neq s_2$  and  $e = 0$ .

Therefore, the VN-processing step of the SCMS decoder can be rewritten as follows:

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do**  $\triangleright$  VN-processing

$\alpha_{m,n}^{(\ell)} = \mathbf{s}_{\mathcal{M}} \left( \mathbf{a}_{\text{pr}} \left( \tilde{\gamma}_n^{(\ell)}, -\beta_{m,n}^{(\ell)} \right) \right);$

$e_{m,n}^{(\ell)} = \text{SCU} \left( s_{m,n}^{(\ell)}, s_{m,n}^{(\ell-1)}, e_{m,n}^{(\ell-1)} \right);$

**if**  $e_{m,n}^{(\ell)} = 1$  **then**  $\alpha_{m,n}^{(\ell)} = 0;$  **end**

This reformulation of the VN-processing step allows defining a *noisy self-correction step*, by injecting errors in the output of the SCU operator. The noisy SCU operator with error probability  $p_{\text{scu}}$  is defined by:

$$\text{SCU}_{\text{pr}}(s_1, s_2, e) = \begin{cases} \text{SCU}(s_1, s_2, e), & \text{with probability } 1 - p_{\text{scu}} \\ 1 - \text{SCU}(s_1, s_2, e), & \text{with probability } p_{\text{scu}} \end{cases} \quad (24)$$

This error model captures the effect of the noisy logic or of the noisy storage of  $s_{m,n}$  and  $e_{m,n}$  values on the SCU operator. The SCMS decoder with noisy self-correction step is detailed in Algorithm 3.

<sup>10</sup>Superscript  $(\ell)$  used to denote the iteration number

---

**Algorithm 3** Noisy Self-Corrected Min-Sum (Noisy-SCMS) decoding
 

---

 Input:  $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$  ( $\mathcal{Y}$  is the channel output alphabet) ▷ received word

 Output:  $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{-1, +1\}^N$  ▷ estimated codeword
**Initialization**
**for all**  $n = 1, \dots, N$  **do**  $\gamma_n = \mathbf{q}(y_n)$ ;

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do**  $\{ \alpha_{m,n} = \gamma_n; s_{m,n} = \text{sgn}(\gamma_n); e_{m,n} = 0; \}$ 
**Iteration Loop**
**for all**  $m = 1, \dots, M$  and  $n \in \mathcal{H}(m)$  **do** ▷ **CN-processing**

$$\beta_{m,n} = \mathbf{x}_{\text{pr}}(\{\text{sgn}(\alpha_{m,n'})\}_{n' \in \mathcal{H}(m) \setminus n}) \mathbf{m}_{\text{pr}}(\{|\alpha_{m,n'}|\}_{n' \in \mathcal{H}(m) \setminus n});$$

**for all**  $n = 1, \dots, N$  **do** ▷ **AP-update**

$$\tilde{\gamma}_n = \mathbf{a}_{\text{pr}}(\{\gamma_n\} \cup \{\beta_{m,n}\}_{m \in \mathcal{H}(n)});$$

**for all**  $n = 1, \dots, N$  and  $m \in \mathcal{H}(n)$  **do** ▷ **VN-processing**

$$\alpha_{m,n} = \mathbf{s}_{\mathcal{M}}(\mathbf{a}_{\text{pr}}(\tilde{\gamma}_n, -\beta_{m,n}));$$

$$e_{m,n} = \mathbf{SCU}_{\text{pr}}(\text{sgn}(\alpha_{m,n}), s_{m,n}, e_{m,n});$$

$$s_{m,n} = \text{sgn}(\alpha_{m,n});$$

**if**  $e_{m,n} = 1$  **then**  $\alpha_{m,n} = 0$ ;

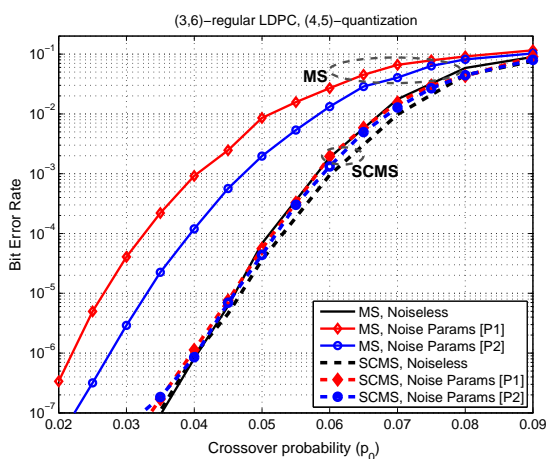
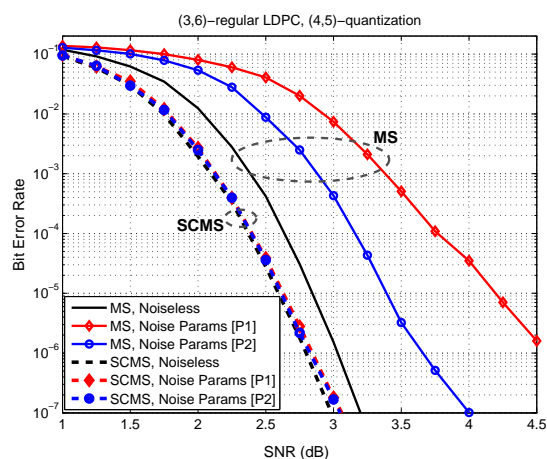
**for all**  $\{v_n\}_{n=1, \dots, N}$  **do**  $\hat{x}_n = \text{sgn}(\tilde{\gamma}_n)$ ; ▷ **hard decision**
**if**  $\hat{\underline{x}}$  is a codeword **then** exit the iteration loop ▷ **syndrome check**
**End Iteration Loop**(a) BSC channel ( $\mu = 6$ )(b) BI-AWGN channel ( $\mu = 5.5$ )

Figure 22. BER performance comparison between noisy MS and noisy SCMS decoders

The finite length performance of the noisy SCMS decoder is presented in Figure 22, for both BSC and BI-AWGN channels. For comparison purposes, Figure 22 also shows the performance of the noisy MS decoder. The parameters of the different noisy components are as follows:

**[P1]** sign-preserving adder with  $p_a = 0.01$ ,  $p_c = 0.01$ ,  $p_x = p_{\text{scu}} = 0.001$  (red curves, diamond markers);

**[P2]** full-depth adder with  $p_a = 0.001$ ,  $p_c = 0.001$ ,  $p_x = p_{\text{scu}} = 0.001$  (blue curves, circle markers).

Solid and dashed curves correspond respectively to the MS and SCMS performance. While the hardware noise alters the performance of the MS decoder, it can be seen that the noisy SCMS decoder exhibits very good performance, very close to that of the noiseless decoder. Therefore, one can think of the self-correction circuit as a *noisy patch* applied to the noisy MS decoder, in order to improve its robustness to hardware noise. The robustness of the SCMS decoder to hardware noise is explained by the fact that it has an intrinsic capability to detect unreliable messages, and discards them from the iterative decoding process [26].

## VIII. CONCLUSION

This paper investigated the asymptotic and finite length behavior of the noisy MS decoder. We demonstrated the impact of the channel scale factor on the decoder performance, both for the noiseless and for the noisy decoder. We also highlighted the fact that an inappropriate choice the channel scale factor may lead to an *unconventional* behavior, in the sense that the noise introduced by the device may actually result in an increased correction capacity with respect to the noiseless decoder. We analyzed the asymptotic performance of the noisy MS decoder in terms of useful regions and target-BER thresholds, and further revealed the existence of a different threshold phenomenon, which was referred to as functional threshold. Finally, we also corroborated the asymptotic analysis through finite-length simulations, and highlighted the excellent performance of the noisy SCMS decoder, which provides virtually the same performance as the noiseless decoder, for a wide range of values of the hardware noise parameters.

## REFERENCES

- [1] M. G. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell System Technical Journal*, vol. 47, pp. 2299–2337, 1968.
- [2] —, "Reliable computation in computing systems designed from unreliable components," *Bell System Technical Journal*, vol. 47, pp. 2339–2366, 1968.
- [3] A. V. Kuznetsov, "Information storage in a memory assembled from unreliable components," *Problemy Peredachi Informatsii*, vol. 9, no. 3, pp. 100–114, 1973.

- [4] S. K. Chilappagari, M. Ivkovic, and B. Vasic, "Analysis of one step majority logic decoders constructed from faulty gates," in *Proc. of IEEE Int. Symp. on Information Theory*, 2006, pp. 469–473.
- [5] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2438–2446, 2007.
- [6] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 56, no. 6, pp. 484–488, 2009.
- [7] Y. Tang, C. Winstead, E. Boutillon, C. Jego, and M. Jezequel, "An ldpc decoding method for fault-tolerant digital logic," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2012, pp. 3025–3028.
- [8] A. M. Hussien, M. S. Khairy, A. Khajeh, A. M. Eltawil, and F. J. Kurdahi, "A class of low power error compensation iterative decoders," in *IEEE Global Telecom. Conf. (GLOBECOM)*, 2011, pp. 1–6.
- [9] L. R. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.
- [10] S. Yazdi, H. Cho, Y. Sun, S. Mitra, and L. Dolecek, "Probabilistic analysis of Gallager B faulty decoder," in *IEEE Int. Conf. on Communications (ICC)*, 2012, pp. 7019–7023.
- [11] S. Yazdi, C. Huang, and L. Dolecek, "Optimal design of a Gallager B noisy decoder for irregular LDPC codes," *IEEE Comm. Letters*, vol. 16, no. 12, pp. 2052–2055, 2012.
- [12] S. Yazdi, H. Cho, and L. Dolecek, "Gallager b decoder on noisy hardware," *IEEE Trans. on Comm.*, vol. 66, no. 5, pp. 1660–1673, 2013.
- [13] A. Balatsoukas-Stimming, C. Studer, and A. Burg, "Characterization of min-sum decoding of LDPC codes on unreliable silicon," in *Information Theory and Applications Workshop (ITA)*, 2014.
- [14] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. PP, no. 99, pp. 1–4, 2014.
- [15] R. G. Gallager, "Low density parity check codes," MIT Press, Cambridge, 1963, research Monograph series.
- [16] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.
- [17] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proc. of the 2nd National Conference on Artificial Intelligence (AAAI-82)*, 1982, pp. 133–136.
- [18] —, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, 1988.
- [19] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Communications*, vol. 47, no. 5, pp. 673–680, 1999.
- [20] S. Chung, "On the construction of some capacity-approaching coding schemes," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.
- [21] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *IET Electronics Letters*, vol. 37, no. 2, pp. 102–104, 2001.
- [22] R. L. Dobrushin and S. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Problemy Peredachi Informatsii*, vol. 13, no. 1, pp. 82–89, 1977.
- [23] A. Amaricai *et al.*, "Circuit level fault models for sub-powered CMOS circuits for uncorrelated and correlated errors," FP7/FET OPEN/309129, i-RISC project, Deliverable D2.1, January 2014. [Online]. Available: <http://www.i-risc.eu>
- [24] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

- [25] D. J. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: <http://www.inference.phy.cam.ac.uk/\discretionary{-}{-}{-}mackay/\discretionary{-}{-}{-}codes/data.html>
- [26] V. Savin, "Self-corrected min-sum decoding of LDPC codes," in *Proc. of IEEE Int. Symp. on Information Theory (ISIT)*, 2008, pp. 146–150.