A fast LDPC encoder/decoder for small/medium codes

J.M.R. Vaza, J.A.B. Gerald

Abstract

Although Low-Density Parity-Check (LDPC) codes perform admirably for large block sizes — being mostly resilient to low levels of channel SNR and errors in channel equalization — real time operation and low computational effort require small and medium sized codes, which tend to be affected by these two factors. For these small to medium codes, a method for designing efficient regular codes is presented and a new technique for reducing the dependency of correct channel equalization, without much change in the inner workings or architecture of existing LDPC decoders is proposed. This goal is achieved by an improved intrinsic Log-Likelihood Ratio (LLR) estimator in the LDPC decoder — the ILE-Decoder, which only uses LDPC decoder-side information gathered during standard LDPC decoding. This information is used to improve the channel parameters estimation, thus improving the reliability of the code correction, while reducing the number of required iterations for a successful decoding. Methods for fast encoding and decoding of LDPC codes are presented, highlighting the importance of assuring low encoding/decoding latency with maintaining high throughput. The assumptions and rules that govern the estimation process via subcarrier corrected-bit accounting are presented, and the Bayesian inference estimation process is detailed. This scheme is suitable for application to multicarrier communications, such as OFDM. Simulation results in a PLC-like environment that confirm the good performance of the proposed LDPC coder/decoder are presented.

1. Introduction

The design and implementation of any contemporary communication system requires trade-offs between system performance, power consumption, reliability and cost. Forward Error Correction (FEC) is one of the available tools for performing such trade-offs. FEC is a method that adds ancillary data as redundancy to a given message. This enables the receiving end to use this ancillary data to detect and/or perform recovery in the event that the transmitted message is corrupted. However, this capacity for reducing transmission error rates by means of adding of any sort of FEC increases overall system complexity and reduces usable bandwidth.

Channel noise and interference from other communications or electronic devices are not the only motivations for the usage of FEC: worldwide telecommunication regulation agencies impose restrictions on the amount of power allowed for certain classes of communications systems. Lower power levels mean higher transmission errors, yet, at the cost of increasing system complexity, FEC can also be used to reduce transmission errors while does comply local power regulations.

The resilience of Orthogonal Frequency-Division Multiplexing (OFDM) [1] to harsh channel conditions such as interference, attenuation and fading makes it a prime choice for modulation in many communication systems. Present-day communication systems employing OFDM are numerous and include broadband wired schemes as ADSL and PLC and wireless schemes as WiFi, IEEE 802.11a/n, WiMAX, DAB, DAB+, HD Radio, T-DMB, ISDB-TSB, and so on. In Table 1 present-day communication systems and services employing OFDM are listed.

Adding FEC to an OFDM system gives rise to what is usually called Coded Orthogonal Frequency-Division Multiplexing (COFDM). Shannon's noisy-channel coding theorem [2] establishes that it is possible to receive an error-free message, sent over an unreliable channel, as long as the information rate does not exceed the channel's capacity. The theorem also assures there is an optimal Error-Correction Code (ECC) scheme, which just adds a minimal redundancy to the original message. Unfortunately, Shannon's theorem is non-constructive: it does not state how to build such an encoder/decoder for (nearly) error-free communications.

Low-Density Parity-Check (LDPC) codes are a very successful family of FEC codes. They are a class of linear block codes that can achieve near channel capacity (0.0045 dB of the Shannon limit [3]), although the feasibility of such codes (and associate decoders)
for real-time applications requiring short codes (low computational effort) can be questioned due to the excessive requirements of computational resources.

LDPC codes were firstly proposed in the 1960s by Gallager [4], but for the most part of the next 30 years those codes remained as forgotten. In the mid 90s these codes were “rediscovered” by different researchers [5], who noticed the very desirable properties of linear block codes with Low-Density (i.e. sparse) Parity-Check matrices (PCM).

LDPC codes are currently employed in communication systems standards such as DVB-S2 [6], 10 Gigabit Ethernet (10GbE) [7], Worldwide Interoperability for Microwave Access (WiMAX) [8], and many others. The majority of deep space missions are using LDPC instead of Reed-Solomon (RS), or other coding mechanisms, since 2000. The OFDM-LDPC association also was the chosen scheme in the G.hn/G.9960 International Telecommunication Union (ITU) standards.

In Power Line Communication (PLC) data is carried over the very noisy power transmission grid, thus requiring good error correction capacity. Broadband over Power Line (BPL) systems deliver broadband services over the typical household power line. Several modulation schemes can be used to transmit data over the PLC channel, but OFDM is well suited to lessen the impact of most of the channel’s conditions such as interference, attenuation and fading, making it a prime choice for the PLC modulation standard (IEEE 1901 standard). HomePlug [9] is one of the major available PLC consumer products. These PLC characteristics make it a suitable environment for testing the low computational effort (i.e. small and medium) LDPC codes and the new schemes proposed in this paper.

Presently, for real time short codes, the turbo codes are the most used ones, relegating the LDPC codes to long code applications where the LDPC codes improved performance can balance the price to pay in terms of their higher computational effort. Nevertheless, several attempts have been made to lighten up short/medium LDPC codes, keeping or even improving their good error correction capacities. These attempts were made either operating on the code generation algorithm [10–13] or on the coder/decoder structure [14–16].

This paper deals with improvements made to an OFDM system with LDPC coding by using a simplified algorithm and a new feedback decoder structure. The remainder of this paper is organized as follows. In Section 2, an LDPC fast encoder is presented and explained. In Section 3, a new improved LDPC decoder for short/medium codes – the Intrinsic Log-Likelihood Ratio Estimator Decoder (ILE-Decoder) – which allows better performance without significant increasing of computational effort, is presented. In Section 4, simulation results in an OFDM PLC-type system are presented that confirm the good performance of the improved encoder/decoder. Conclusions and comments are addressed in Section 5.

2. LDPC fast encoder

2.1. Fast encoder

Several parameters determine the performance of LDPC codes regardless any other implementation details or the system’s operating conditions. But even a code designed for the utmost coding performance must also have efficient run-time encoding and decoding methods, otherwise, it is not suitable for real-time and/or low-latency usage. Dozens of LDPC building methods or improvements to existing methods continue to be proposed annually, as for instance, cyclic geometries, finite-geometries [17], progressive edge growth and its many variants [18], combinatorial constructions [19], and so on. Among most of these proposals introduce gains in the efficiency of LDPC error correcting, these gains are at the expense of run-time performance and/or system complexity.

Successful applications of LDPC systems tend to prefer simpler codes with simple thoroughly tested encoder/decoder structures, such as the LDPC code used in 10GBase-T Ethernet, where the validation and optimization of that code took 2 weeks in a 256 node computer cluster [7]. LDPC codes can be of two different classes regarding their code graph representation: regular and irregular. Regarding hardware implementations, for irregular codes, where different nodes can have a varying number of incident edges, this “extra” variety increases the number of hardware resources needed at the decoder, and, it also puts extra demands on the scheduling algorithms that sequences the Message-Passing Algorithm (MPA). Conversely, regular codes only need to have one type of hardware block to compute check equations. Thus, henceforth we will only consider regular codes.

For dense matrices, an n sized vector-matrix multiplication has complexity O(n^2). Simplifications can be made, but these simplifications and/or restrictions in the code structure always result in loss of coding performance [3]. Richardson and Urbanke in 2001 [20] introduced a method that allowed efficient encoding of LDPC codes in linear time, with the provable maximum complexity of:

\[
C(n) = 0.0172n^2 + O(n) \tag{1}
\]

In linear block codes, a message (a sequence of K user bits, \( u=[u_1, u_2, \ldots, u_K] \)) is encoded through linear block coding into an N bit sequence, called a codeword, \( x=[x_1, x_2, \ldots, x_N] \). The linear block encoding can be mathematically expressed as \( x = G u \) where \( G \) is a specially crafted matrix (generator matrix) which has the rules for adding redundancy to \( u \). As a linear-block code, an LDPC code has a parity-check matrix (PCM) representation, denoted by an \( n \times m \) matrix \( H \). The LDPC algorithm has two parts: an offline (preprocessing) and an online part. In the preprocessing part of the algorithm, the matrix \( H \) is firstly rearranged into an approximate triangular form suitable for fast encoding. This is achieved by iteratively permuting the row/columns until a suitable partition of \( H \) is found. It is performed by rearranging and partitioning the \( H \) matrix with a gap parameter, \( g \), as small as possible, since the most expensive step is the dense matrix multiplication with complexity \( C(g \times (n-m)) \). Thus, considering (1), the encoding process complexity is now set by \( g \). Another advantage of this encoding process is storage savings, since there is no need to store the generator matrix \( G \), because a full encoder/decoder system can perform both operations with only the parity-check matrix \( H \).

The flowchart in Fig. 1 presents the order of the steps required to achieve a well performing code. Previously, from the input

Table 1

<table>
<thead>
<tr>
<th>Usage</th>
<th>System/Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadband</td>
<td>ADSL and PLC (both wired)</td>
</tr>
<tr>
<td>Computer networks</td>
<td>WiFi, IEEE 802.11a/n and WiMAX</td>
</tr>
<tr>
<td>Terrestrial TV</td>
<td>DVB-T, DVB-H, T-DMB, ISDB-T and MoCA</td>
</tr>
<tr>
<td>Satellite TV</td>
<td>DVB-S2</td>
</tr>
<tr>
<td>Cellular communication</td>
<td>Flash-OFDM</td>
</tr>
<tr>
<td>Personal Area Networks</td>
<td>UWB</td>
</tr>
<tr>
<td>Mobile broadband</td>
<td>3GPP and HSDPA</td>
</tr>
</tbody>
</table>
information \((n; r; j; k)\) — (block length, code rate, column weight, row weight) — the LDPC code generator algorithm starts by obtaining the input parameters \((n; m; j; k)\), where the number of constraints \(m\) is derived from the input parameters \(n\) and \(r\) (block size and code rate). Then the first step is to create an empty \(m \times n\) matrix. For irregular codes, \(j\) and \(k\) contain the desired distribution of 1s by column and row respectively. For regular codes, \(j\) and \(k\) cardinality is 1.

The algorithm employed for distributing the 1s throughout the parity-check matrix was a random opportunistic greedy cardinality is 1. The small loops removal is essential for a code to remove its graph. After most of the small loops are removed, the process continues into the next phase. Otherwise, the process restarts with a new empty matrix. As the small loops removal is essential for a code with good performance, this procedure is detailed in the next section.

The next step in the building process, to generate a fast encoding representation from the current code, is also responsible for the existence of a fast encoding process. This step is implemented according to the ideas of [20].

As a remark, the Progressive Edge Growth (PEG) method for building low-density parity-check matrices could also have been tried here. The supposed advantage of such method is building in one single procedure \(H\) matrices that are guaranteed to have a fast encoding representation while having high girths. But as disadvantages, this method includes heavy computational burden upon code generation (possibly weeks [7]), and it may not prevent a deviation from the code’s desired input parameters.

### 2.2. Short cycles removal

Some parameters and properties can greatly affect a code’s performance, as for instance the girth size and code structure. The girth is defined as the smallest loop/cycle (the path from a given node back to itself) within a code’s graph representation. A code’s minimum girth is the parameter that mostly affects performance, particularly for small-sized codes. Smaller cycles, (see Fig. 2) mean that less reliability for a given bit can be gathered by the decoder. Because codes have finite block-length, any cycles on its graph are bounded. Thus, at every node there are many possible paths that revert back to the original node. Messages over larger cycles take more iterations to travel back to a given starting node, but have the opportunity to gather reliability updates from more distant nodes. For small cycles the reverse is true: spanning the entire cycle takes less iterations, but fewer nodes are reached, thus biasing the reliability updates of their members.

Most authors [21] acknowledge the desirability of removing small loops in LDPC graphs, due to the loss of error coding performance caused by such small cycles. With some shuffling and row/column swapping it is usually possible to remove a 4-cycle. Longer cycles, are not as easy. Approaches such as those given in [22], which just delete small cycles, tend to deeply change the properties of a LDPC code. An effective small loop removal process must perform and assure the following two conditions: (i) to be able to locate all nodes with a girth of size \(n\); (ii) to be able to remove a \(n\)-loop without introducing smaller loops. In order to efficiently locate loops of any length, the removal process is performed over a code’s adjacency matrix. Because the PCM of LDPC codes represents a bipartite graph, the adjacency matrix \(A\) is simply:

\[
A = \begin{bmatrix} 0 & H \\ H^T & 0 \end{bmatrix}
\]

where 0 is the all-zero matrix. As presented in [23], the following observations regarding \(A\) stand:

**Theorem 1.** The \((i; j)\)-entry of \(A^2\) is the number of paths of length \(n\) from \(v_i\) to \(v_j\)

**Theorem 2.** In a graph with girth \(n\), two nodes \(v_i\) and \(v_j\) are opposite each other (i.e. connected via a 3rd node) if:

\[
A_{ij}^{n/2} \geq 2
\]

\[
A_{ij}^{n/2-2} = 0
\]

With Theorem 2, it is possible to identify which nodes have a particular girth. Once an unwanted loop is found, the next step is to remove it from the graph.

Consider the small loop (4-cycle) depicted in the Fig. 2, spanning the vertexes \(c_1, v_3, c_2\) and \(v_3\): at variable node \(v_3\), there can be only reliability updates coming from \(c_1\) and \(c_2\). After 2 iterations

**Fig. 1.** Flowchart for finding high-performing LDPC codes.

**Fig. 2.** A 4-loop defined by vertexes \(v_3, v_3, c, v_3\), in both tanner graph (right) and parity-check matrix forms (left). A 6-loop defined by vertexes \(v_3, c, c_2, c_1, c_2, c_1, v_3\).
(two “up” and two “down” half-iterations), \( v_3 \) will receive updated reliabilities weighting the whole \( v_3c_1v_7c_3 \) cycle. There are of course larger cycles that also have \( v_3 \) as a vertex (see the dashed 6-cycle \( v_3c_1v_7c_3v_2c_5 \)). But the consequence is that every 2 iterations \( v_3 \) will receive through \( c_1/c_3 \) similar updates, weighting mostly \( c_1/c_3 \). Messages from distant nodes will arrive at a slower pace: reliabilities over the whole dashed 6-cycle will arrive every 3 iterations. Thus, the reliability updates received at \( v_3 \) will be heavily biased towards the information updates that happen over the small 4-cycle. This is a limiting result: the condition of convergence of the MPA assumes that the cycles spanning at each node are infinite.

The typical algorithm for generating LDPC codes usually makes a minimal effort to discard at least 4-cycles, because they are easy to spot in a parity-check matrix: search for columns with two 1s in identical positions (thus forming a rectangle of four 1s in the matrix — see Fig. 2). Longer cycles, are not as easily removed as 4-cycles.

Succinctly, the removal process is as follows: in order to efficiently remove a small loop, a random edge from that loop is selected, with nodes \( v_1 \) and \( v_2 \). From the set of the remainder adjacent nodes of \( v_1 \), a node \( v_3 \) is selected giving rise to a new edge \( e=E[v_1v_3] \), where \( E[.] \) denotes the average operator. Then, from the set of all nodes with a distance greater than \( n-1 \) from both \( v_1 \) and \( v_3 \), a random edge \( v_4 \) is selected. From the previous set \( S_n \), an edge \( f=E[v_4v_n] \) with both ends in that set is selected. The actual loop removal step, deletes both \( e \) and \( f \) from the graph, and replaces them with \( e_2=E[v_1v_n] \) and \( f_2=E[v_3v_n] \). The details and the needed cautionary steps required to successfully perform a loop removal are detailed in [23].

3. Improved LDPC decoder

This section introduces an architecture — the ILE — Decoder — that delivers improvements to the decoding stage of multicarrier LDPC-coded systems, without resorting to high-complexity LDPC implementations, or increasing the complexity of existing ones. It uses the bit information present at the end of an LDPC decoder, to build an accurate model of the channel noise.

From a generator matrix \( G \), a PCM \( H \) can be derived, where \( H=\left[-P^T|I_{k-m}\right] \), if and only if the \( G \) matrix has a decomposition such as \( G=I_{k-m}P \). The \( H \) matrix has all the verification rules that a given codeword must satisfy in order to be declared error free [4]. For linear block codes, satisfying the condition \( xH=0 \) proves that a codeword is valid and error-free. This property, however, allows only detecting errors, not correcting them. In order to correct errors, one can apply methods such as the algebraic syndrome decoding method, or a probabilistic method such as the Viterbi Algorithm (VA) [24], which minimizes the probability of decoding error in a given codeword. Although the VA decoding was conceived for convolutional decoding, the work by [25] showed that soft decoding techniques of convolutional decoding could be applied to block codes. This brought the power of maximum likelihood decoding to block codes, because the algebraic decoding, although quite powerful, only performs well in low-noise/linear channels.

Relating to the LDPC case, regarding the challenge of implementing a simpler decoder, Gallager introduced a near-optimal decoding [26] algorithm on which our approach is based. The decoding algorithm has different names, since it was “invented” several times in distinct domains: Sum-Product Algorithm (SPA), the Belief-Propagation Algorithm (BPA), and the Message-Passing Algorithm (MPA).

OFDM systems already split modulated data through several sub-bands. Assuming a frequency-selective noise environment, typical of PLC or DSL (Digital Subscriber Line), this paper introduces a modification for the Binary Symmetric Channel (BSC) that takes into account frequency-selective noise. The standard BSC channel is composed into a bank-like structure, where each subcarrier, is a single standard BSC, as depicted in Fig. 3.

Each subchannel \( 1, 2, \ldots, N \), has its own probability of cross-over, \( p_{1,1}, p_{2,2}, \ldots, p_{N,N} \), that is flipping from 1 to 0 or from 0 to 1. In this stacked model, each single BSC is intended to model a particular frequency of frequency-selective noise. Because of the modifications made to the LDPC decoder, it is irrelevant where in the electromagnetic spectrum the first and subsequent subcarriers are located, how each one is spaced, interleaved, what was the modulation type, etc. The decoder is only concerned with the provability of every bit and through which channel did a particularly bit come through. The reasoning is that each subcarrier is essentially sampling the noise of that band of spectrum. Thus, all uncorrected bits coming from that subcarrier are sampled in the same sub-band of the spectrum, regardless of the modulation scheme. As long as such bit vs. channel accounting is possible the approach described in this paper is applicable.

Using Bayes’ Theorem, the probability of an input \( x \) given the received codeword \( y \), is:

\[
P(x|y) = \frac{P(y|x)P(x)}{P(y)}
\]  

(5)

For LDPC decoding, the most efficient known method for computing each of the \( x_k \) bits is the a posteriori probability (APP), \( P(x_k=1|y) \), computed through the Message-Passing Algorithm (MPA). It iteratively computes the probability of a transmitted codeword bit \( x_k \) being 1, given all received codeword’s bits \( y=[y_0, y_1, \ldots, y_{N-1}] \), that is \( P(x_k=1|y) \). Thus, following APP ratio can be used:

\[
x_k=\frac{P(x_k=1|y)}{P(x_k=0|y)}
\]  

(6)

Or, for better stability, the Log-Likelihood Ratio (LLR):

\[
\lambda_{x_k}=\ln(x_k)=\ln\left(\frac{P(x_k=1|y)}{P(x_k=0|y)}\right)
\]  

(7)

[Fig. 3. Example of multichannel noisy Binary Symmetric Channel model.]
The MPA is an iterative algorithm based on the code’s graph for the computation of $P(x=1|y)$, $\ln(x_n)$ or $\lambda_n$. The decoder operates as follows: (i) iteratively, the decoder evaluates the codeword’s check-constraints; (ii) it notifies neighboring nodes of the confidence level of that bit being correct; (iii) each node, given the new incoming confidence level, recalculates its own confidence levels; (iv) again, it notifies neighboring nodes. This confidence propagation occurs through all edges, from $c$-nodes to $v$-nodes, and back again, several times, until a predefined number of iteration is reached, or the codeword is valid. The estimate, $x_{n,i}$, of a given bit being 0 or 1 in the log-likelihood domain is given by:

$$
\begin{cases}
    \lambda_{n,i} > 0 \Rightarrow x_{n,i} = 1 \\
    \lambda_{n,i} < 0 \Rightarrow x_{n,i} = 0
\end{cases}
$$

(8)

Thus, the estimate that a received bit being 0 or 1 depends only on the sign of $\lambda_{n,i}$, $x_{n,i}=\text{sign}(\lambda_{n,i})$. Because the computation of the log-MAP via the MPA (computing the estimate of each $x_n$ value), occurs on a graph, two sources of reliability are available: the reliability of $x_n$ given $y_n$, and the reliability of $x_n$ given all the bits of $y$ except $y_n$ that is: $y_{n,\text{ex}}$. Taking into account these sources of reliability and Bayes’ theorem in both the numerator and denominator of (7):

$$
\lambda_{n,i} = \ln \left( \frac{P(y_n|x_n=1,y_{n,\text{ex}})}{P(y_n|x_n=0,y_{n,\text{ex}})} \right) = \ln \left( \frac{P(y_n|x_n=1,y_{n,\text{ex}})}{P(y_n|x_n=0,y_{n,\text{ex}})} \right) + \ln \left( \frac{P(x_n=1|y_n)}{P(x_n=0|y_n)} \right)
$$

The received channel’s sample $y_n$ is independent of the set of received samples $(y_{n,\text{ex}})$. Thus, (9) simplifies to:

$$
\lambda_{n,i} = \ln \left( \frac{P(y_n|x_n=1)}{P(y_n|x_n=0)} \right) + \ln \left( \frac{P(x_n=1|y_n)}{P(x_n=0|y_n)} \right)
$$

(11)

The first term of (11), contains the intrinsic information that is the reliability of $x_n$ based on the channel sampling $y_n$. The second term contains the extrinsic information, produced by the decoder using all channel samples except the current bit $n$. In our work, we deal with improvements to the LDPC decoding process, in what relates to channel noise estimation. The typical LDPC decoder block diagram is shown in Fig. 4.

![LDPC decoder block diagram](image)

Fig. 4. LDPC decoder block diagram.

From Fig. 4 we can see that at the noisy channel end, each bit measure, $y_n$, plus associated noise, $\varepsilon_n$, is assigned a log-probability ratio, LLR, $\lambda_{n,\text{int}}$. This intrinsic LLR quantity is the only information/measurement available at the end of the noisy channel, and it is this information that enters the LDPC decoder. Inside the decoder, the iterative process Message-Passing Algorithm (MPA) takes place, which will check and infer whether the reliability of a given bit (given all other bits measured at the channel’s exit) accurately represents a given bit. At each iteration the degree of confidence increases by successively harvesting all extrinsic LLRs. After a preset number of iterations, a hard-decision is performed yielding the estimated decoded codeword $\hat{y}_n$. The intrinsic LLR quantity is a function of two parameters: a channel bit observation $y_n \in \{0, 1\}$, and channel noise conditions. For instance, for the additive white Gaussian noise (AWGN) channel the intrinsic LLR for bit $x_n$ can be found as follows. Being the noise a normal random variable with zero mean and variance $\sigma^2$, at the output we have:

$$
P(y_n|x_n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_n-x_n)^2}{2\sigma^2}}
$$

(12)

The received channel’s sample $y_n$ is independent of the set of received samples $(y_{n,\text{ex}})$. Thus, (9) simplifies to:

$$
\lambda_{n,i} = \ln \left( \frac{P(y_n|x_n=1)}{P(y_n|x_n=0)} \right) + \ln \left( \frac{P(x_n=1|y_n)}{P(x_n=0|y_n)} \right)
$$

(10)

Intrinsic
Extrinsic

Table 2

<table>
<thead>
<tr>
<th>Code</th>
<th>Number of n-cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>276</td>
</tr>
<tr>
<td>$H_2$</td>
<td>10</td>
</tr>
<tr>
<td>$H_3$</td>
<td>1252</td>
</tr>
<tr>
<td>$H_4$</td>
<td>2098</td>
</tr>
<tr>
<td>$H_5$</td>
<td>984</td>
</tr>
<tr>
<td>$H_6$</td>
<td>56</td>
</tr>
</tbody>
</table>

Fig. 5. ILE-Decoder inner block structure.

![ILE-Decoder inner block structure](image)

Fig. 5. ILE-Decoder inner block structure.

![Overall system architecture](image)

Fig. 6. Overall system architecture.
It is now possible to calculate the intrinsic information for bit $x_n$:

$$\lambda_{x_n,\text{int}}(y_n, s^2) = \ln \left( \frac{P(y_n|x_n = 1)}{P(y_n|x_n = 0)} \right) = \ln \left( \frac{\exp(-\frac{(y_n-1)^2}{2s^2})}{\exp(-\frac{(y_n+1)^2}{2s^2})} \right)$$

As evidenced by Eq. (13), both the bit observation $y_n$ and the AWGN channel noise parameter, $s^2$, are the parameters that define the intrinsic LLR. Different channels types have different expressions for the LLRs, which mirrors that particular channel noise model. Thus, any uncertainty in the estimate of the channel noise parameter will also be reflected as uncertainty in the intrinsic LLR value. As shown in [27], intrinsic LLR values that result from estimating the channel noise as greater than the real noise value, do not affect the error correction code, but intrinsic LLRs computed over an optimistic noise parameter (lower than the real noise value) decrease the correction ability. Also, the intrinsic LLR needs not to be modified/recalculated unless a change in the channel noise properties occurs.

### 3.1. LDPC ILE-Decoder

The unknown parameter of interest here is the channel noise. The available/measured data is the data at the input and output of the decoder. By comparing how many bits are corrected between the LDPC decoder input and output, it is possible to infer the channel noise, and thus compute a better estimate for the intrinsic LLR. This means, for the BSC, estimating $p_c$—the probability of

![Fig. 7. $H_1$ vs. $H_2$ with girth removal step applied.](image)

![Fig. 8. Posteriori evolution of an error process with $p_{c_1}=0.1$ (dashed line) and $p_{c_2}=0.4$ (solid line).](image)
cross-over, and, for the BPSK-AWGN channel, estimating the variance, $s^2$. In a multichannel case, this method must be applied to each subchannel separately. For the Binary Symmetric Channel (BSC), there are only two possible outcomes for a received bit $y$, given a channel input $x$: this is Bernoulli trial, which has a binomial distribution model, and the probability mass function (pmf) of the binomial distribution is:

$$f(k;n,p) = \binom{n}{k}p^k(1-p)^{n-k}$$  \hspace{1cm} (14)

where $n$ is the number of trials, $k$ successes (here “success” is the event of cross-over), and $p$, the estimated parameter.

As a remark, for the BPSK with AWGN channels a solution is to use the Normal approximation to the binomial distribution [28], where instead of the usual $\mathcal{N}(\mu; \sigma^2)$ there is the approximation $\mathcal{N}(\mu=np; \sigma^2=np(1-p))$. This broad interpretation means that the BPSK AWGN channel, from the point of view of the LDPC decoder is also a BSC bank channel. It is worth noting that other methods for estimating the variance may be derived, such as finding the inverse Gaussian-like distribution, or even finding a closed form expression via the Bayes Theorem, by taking into account the methods presented in [28]. This undertaking, for its complexity and size, deserves further study.

Since the underlying probability distribution process is known (see [14]), methods such as Bayesian inference can be used to find a better and faster estimate of the channel noise parameter. The Bayes theorem, used herein as the method for estimation, is as follows:

$$P(X|Y,I) = \frac{P(Y|X,I) \times P(X|I)}{P(Y|I)}$$  \hspace{1cm} (15)

Now, naming the variables as hypothesis, data, and $I$ (information), the data analysis and inference capability becomes [28]:

$$P(\text{hypothesis}|\text{data},I) = \frac{P(\text{data}|\text{hypothesis},I) \times P(\text{hypothesis}|I)}{P(\text{data}|I)}$$  \hspace{1cm} (16)

Regarding iterative estimation, the general estimation procedure for Bayesian estimation has the following form:

$$P(\text{new hypothesis}) = \frac{P(\text{previous hypothesis}) \times P(\text{new data})}{\text{normalization factor}}$$  \hspace{1cm} (17)

The ILE-Decoder block diagram is shown in Fig. 5. Again, the core of the ILE-decoding algorithm is the MPA. The MPA iteratively tries to gather and increase the assurance that a particular bit is either a one or zero.

![Fig. 9. Posteriori evolution of an error process with a $p_c = 0.30$.](image)
In the initial decoding iteration, the MPA is supplied with an intrinsic LLR that measures the reliability of a given bit being a one or zero given the channel conditions $y_n + e_n$. This LLR is then added (or composed) to the extrinsic LLR computed by the MPA in an iterative fashion. After a preset number of iterations, a hard-decisioning is performed on the extrinsic LLR values yielding a decoded (and/or corrected) codeword $\hat{y}_n$. The proposed addition is an “Intrinsic LLR estimator” that, through the account of which bits were modified on a given subchannel, tries to infer subchannel noise parameters, and thus compute a more reliable value for the intrinsic LLR that enters the LDPC decoder in order to improve the decoder error correction performance.

### 4. System performance

The overall system architecture, encoder, decoder, modulator, demodulator, channels, but without the subchannel error estimation block discriminated, is presented in Fig. 6. The operation of this system is as follows: consider the five bit message $u$ (top left corner — represented as numbers inside circles). These are encoded into an LDPC codeword, $x$ (bits 1 to 7) where the boxes 6 and 7 are the redundancy bits introduced by the LDPC encoder. Then, these bits are introduced into a multichannel modulator (while being conveyed from serial to parallel ordering) where each subchannel can have different noise profiles (modulated bits represented by triangles). In the receiver the bits are demodulated and serialized into an estimated codeword $y$, which is sent to the LDPC decoder, yielding at the output an estimate for the original message $u$.

In order to show the detrimental effect of small cycles, a random LDPC code ($n=140$; $m=35$; $j=3$; $k=12$) with Parity-Check Matrix (PCM) $H_1$ is introduced. $H_2$ is the PCM of the same code but with most of the small cycles removed by the removal process described previously. Table 2 details the girth removal algorithm results. For each code (first column), its number of $\{4, 6, 8\}$-cycles are detailed.

It is easily noticeable from Table 2 that there are much less 4-cycles in $H_2$ than in the original PCM $H_1$. Notice that the total number of remaining cycles before, and after, the removal of small cycles is not constant due to the way how the removal process works, merging smaller cycles into larger ones, thus decreasing the total number of cycles in a graph.

Fig. 7 represents the Bit Error Rate (BER) vs. SNR for a simulation performed over the AWGN channel. It is clear the lower BER presented by $H_2$ when compared with that of $H_1$.

For greater clarity, three examples of iterative unfolding of the estimation process were depicted into two figures, namely Figs. 8 and 9. For simplicity, only the first 9 iterations (a)–(i) of the convergence process are shown in both figures. Fig. 8 shows the results for two subcarriers with channel noises $p_{c1} = 0.1$ and $p_{c2} = 0.2$; Fig. 9 shows the results for one subcarrier with channel noises $p_c = 0.3$.

The estimation process starts with “full ignorance”, the estimated parameter could be uniformly distributed in the whole 0–1 probability interval. As the process iterates and new information is available, it starts to bend and narrow the range where the estimated parameter might lie. As Figs. 8 and 9 show, after few iterations the maximum of the pdf is already located near the real values.

To measure the performance gains of the estimator block, two runs were performed: run 1 with the augmented LLR estimator (ILE-Decoder) and run 2 with just a plain multicarrier LDPC coded system. In order to dismiss any performance differences caused by the random generation of source messages $u$ and the random generation of channel noise, both runs were performed with pre-generated batches of source message and BSC noise, and these two batches were replayed for each test run. The LDPC code employed was an ($n=140$; $m=35$; $j=3$; $k=12$). As can be seen from Table 3, in run 1 both the total number of iterations and total number of errors are reasonably lower than in run 2.

### 5. Conclusions

In this paper methods for improving an LDPC encoder and decoder, were presented. Small to medium LDPC codes, which tend to perform good enough while keeping the system complexity low, were targeted. A fast encoder (for real time processing) and an improved decoder — the ILE-Decoder — were introduced. The main change in the decoder was the addition of an intrinsic LLR estimator to the LDPC decoder, in order to better estimate each subchannel noise parameters. The concept of estimating and then applying that estimation was detailed. The assumptions and rules that govern the estimation process via subcarrier corrected-bit accounting were presented, and the Bayesian inference estimation process was detailed.

As shown, it is possible to reduce the number of iterations taken for decoding a valid LDPC codeword, while slightly increasing the error recovery rate of existing systems. Simulation results for the new coder/decoder were presented, which confirm its better performance regarding the classic structures.

The proposed solutions are suitable for any application with multicarrier communications, such as OFDM. The PLC was chosen as system environment but the results can apply to other medium requiring small to medium LDPC codes.

### Acknowledgments

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under Project Pest-EE/UI0110/2011.

### References


