A New LDPC Coder/Decoder for PLC

José M. R. Vaz 
INESC-ID/IST 
Lisbon, Portugal 
e-mail: jmgm@sips.inesc-id.pt

José A. B. Gerald 
INESC-ID/IST 
Lisbon, Portugal 
e-mail: jabg@inesc-id.pt

Abstract— Adding the ability of LDPC to an OFDM system gives rise to a robust and suitable technique for broadband PLC. However, despite LDPC codes performing admirably for large block sizes, real-time operation and low computational effort require small and medium-sized codes, which tend to be affected by channel SNR and errors in channel equalization. This paper deals with improvements made to an OFDM system for PLC, when using small/medium LDPC coding. Methods for fast encoding and decoding of LDPC codes are presented, highlighting the importance of assuring low encoding/decoding latency with maintaining high throughput. In the receiver, an improved intrinsic Log-Likelihood Ratio (LLR) estimator to the LDPC decoder – the ILE-Decoder – is presented, which better estimates each subchannel noise parameters. The assumptions and rules that govern the estimation process via subcarrier corrected-bit accounting are presented, and the Bayesian inference estimation process is detailed. Simulation results in a PLC environment that confirm the good performance of the proposed LDPC coder/decoder are presented. At the end conclusions and final comments are addressed.

Keywords- LDPC; OFDM; PLC;

I. INTRODUCTION

Forward Error Correction (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 the transmitted message is corrupted, although at the expenses of increasing system complexity and reducing usable bandwidth. Channel noise and interference from other communications or electronic devices are not the only sources that push for the usage of FEC: worldwide telecommunication regaition agencies impose restrictions on the amount of power allowed for certain classes of communications systems. Lower power levels means higher transmission errors, thus, at the cost of increasing system complexity, FEC can also be used to reduce transmission errors, while complying local power regulations.

Several modulation schemes can be used to transmit data over the Power Line Communication (PLC) channel, but Orthogonal Frequency-Division Multiplexing (OFDM) [1] is well suited to lessen the impact of most of the channel’s conditions such as interference, attenuation and fading. Adding FEC to an OFDM system gives rise to what is usually called Coded Orthogonal Frequency-Division Multiplexing (COFDM). In Table I present-day communication systems and services employing OFDM are listed.

LDPC codes are a very successful family of FEC codes. LDPC codes are a class of linear block codes that can achieve near channel capacity (0.0045dB of the Shannon limit [2]), although the feasibility of such codes (and associate decoders) for real-time applications can be questioned due to the excessive requirements of computational resources. LDPC codes are currently employed in communication systems standards such as DVB-S2 [3], 10 Gigabit Ethernet (10GbE) [4], Worldwide Interoperability for Microwave Access (WiMAX)[5], 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.

Power Line Communication (PLC) is a class of systems where data is carried over the very noisy power transmission grid. Broadband over Power Line (BPL) systems deliver broadband services over the typical household power line. The robustness of OFDM to severe channel conditions makes it an evident choice as the standard (IEEE 1901 standard) for modulation in PLC systems. HomePlug [6] is one of the major available PLC consumer products.

This paper deals with improvements made to an OFDM system for PLC when using the Low-Density Parity-Check (LDPC) coding. In Section II of this paper a LDPC fast Encoder is presented and explained. In Section III, an 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 IV simulation results are presented that confirm the good performance of the improved coder/decoder; the ILE-Decoder reduces the number of iterations taken by the LDPC decoder in short/medium codes transmission. In Section V conclusions and comments are addressed.

This work was supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

I. INTRODUCTION

Forward Error Correction (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 the transmitted message is corrupted, although at the expenses of increasing system complexity and reducing usable bandwidth. Channel noise and interference from other communications or electronic devices are not the only sources that push 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 means higher transmission errors, thus, at the cost of increasing system complexity, FEC can also be used to reduce transmission errors, while complying local power regulations.

Several modulation schemes can be used to transmit data over the Power Line Communication (PLC) channel, but Orthogonal Frequency-Division Multiplexing (OFDM) [1] is well suited to lessen the impact of most of the channel’s conditions such as interference, attenuation and fading. Adding FEC to an OFDM system gives rise to what is usually called Coded Orthogonal Frequency-Division Multiplexing (COFDM). In Table I present-day communication systems and services employing OFDM are listed.

Table I: List of Present-day Communication Systems Employing OFDM

<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 HSOPA</td>
</tr>
</tbody>
</table>

LDPC codes are a very successful family of FEC codes. LDPC codes are a class of linear block codes that can achieve near channel capacity (0.0045dB of the Shannon limit [2]), although the feasibility of such codes (and associate decoders) for real-time applications can be questioned due to the excessive requirements of computational resources. LDPC codes are currently employed in communication systems standards such as DVB-S2 [3], 10 Gigabit Ethernet (10GbE) [4], Worldwide Interoperability for Microwave Access (WiMAX) [5], 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.

Power Line Communication (PLC) is a class of systems where data is carried over the very noisy power transmission grid. Broadband over Power Line (BPL) systems deliver broadband services over the typical household power line. The robustness of OFDM to severe channel conditions makes it an evident choice as the standard (IEEE 1901 standard) for modulation in PLC systems. HomePlug [6] is one of the major available PLC consumer products.

This paper deals with improvements made to an OFDM system for PLC when using the Low-Density Parity-Check (LDPC) coding. In Section II of this paper a LDPC fast Encoder is presented and explained. In Section III, an 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 IV simulation results are presented that confirm the good performance of the improved coder/decoder; the ILE-Decoder reduces the number of iterations taken by the LDPC decoder in short/medium codes transmission. In Section V conclusions and comments are addressed.

This work was supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.
II. LDPC FAST ENCODER

Successful applications of LDPC systems tend to prefer simpler codes with simple encoder/decoder structure though thoroughly tested, 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 [4]. LDPC codes can be of two different classes regarding the 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 [2]. Richardson & Urbanke in 2001 [7] introduced a method that allowed efficient encoding of LDPC codes in linear time, with the provable maximum complexity of:

$$C(n) = 0.0172n^2 + O(n)$$  \hspace{1cm} (1)

In linear block codes, a message (a sequence of $K$ user bits, $u = [u_0 u_1 \ldots u_{K-1}]$) is encoded through linear block coding into an $N$ bit sequence, called a codeword, $x = [x_0 x_1 \ldots x_{N-1}]$. The linear block encoding can be mathematically expressed as $x = uG$ 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 $H$ of $n$ by $m$ dimension. 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 a rearranging and partition the $H$ matrix with a gap parameter, $g$, as small as possible, because the most expensive step is the dense matrix multiplication with complexity $O(g \times (n - m))$. Thus, as stated in (1), the encoding process complexity is set by $g$. Another advantage of this encoding process is storage savings: 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 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 $m$ by $n$ matrix. For irregular codes, $j$ and $k$ contain the desired distribution of 1’s by column and row respectively. For regular codes, $j$ and $k$ cardinality is 1.

![Flowchart for finding high-performing LDPC codes.](image-url)

The algorithm employed for distributing the 1’s throughout the parity-check matrix was a random opportunistic greedy filling algorithm that given a target row/column weight distribution $(j; k)$ or $(h; l)$, tries on each iteration to fill still vacant rows/columns, while avoiding violating weighting distribution rules. After the filling process ends, the code must be checked for compliance with the input parameters. Non-compliant codes are discarded and the process restarts.

The next phase in the process tries to break small loops into larger loops. The smallest loop in a LDPC code/graph is a loop with 4 cycles. If most of the small loops were removed, the process continues into the next phase. Else, the process restarts with a new empty matrix. Some parameters and properties can greatly affect a code 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 iteration, and fewer nodes are reached. Thus, smaller cycles biases incoming messages of all its spanning nodes towards the reliability updates of the cycles members.

Consider the small loop (4-cycle) depicted in the Fig. 2, spanning the vertexes $c_1$, $c_2$, $c_3$, and $v_1$. At variable node $v_1$, there can be only reliability updates coming from $c_1$ and $c_3$. After 2 iterations (two “up” and two “down” half-iterations),
v_3 \text{ will receive updated reliabilities weighting the whole } v_3c_1v_5c_2v_1c_3 \text{ cycle. There are of course larger cycles that also have } v_3 \text{ as a vertex (see the dashed 6-cycle } v_3c_1v_7c_3). \text{ But the consequence is that every 2 iterations } v_3 \text{ will receive through } c_1/c_3 \text{ similar updates, weighting mostly } c_1/c_3. \text{ Messages from distant nodes will arrive at a slower place: reliabilities over the whole dashed 6-cycle will arrive every 3 iterations. Thus, the reliability updates received at } v_3 \text{ 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, aren’t as easily removed as 4-loops. Approaches such as given in [8], which just delete small cycles, tend to deeply change the LDPC code properties. The details and the needed cautionary steps required to successfully perform a loop removal are detailed in [9].

### III. IMPROVED LDPC DECODER

This section introduces an architecture – the ILE-Decoder – that delivers improvements to the decoding stage of multcarrier 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 \(H = [\mathbf{P}^T \mid \mathbf{I}_{(n-k)}]\), if and only if the \(G\) matrix has a decomposition such as \(G = [\mathbf{I} \mid \mathbf{P}]\). The \(H\) matrix has all the verification rules that a given codeword must satisfy in order to be declared error free [10]. For linear block codes, satisfying the condition \(xH^T = 0\) 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) [11], which minimizes the probability of decoding error in a given codeword. Although the VA decoding was conceived for convolutional decoding, the work by [12] 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 [13] algorithm we can base on. OFDM systems already split modulated data through several sub-bands. In this paper, we introduce 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, . . . , N, has its own probability of cross-over, \(p_{c1}; p_{c2} \ldots p_{cN}\), 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 provenance of every bit and through what 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 sampling 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.

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

\[
\ln(P(x_n = 1 \mid y)) = \frac{P(x_n = 1 \mid y)}{P(x_n = 0 \mid y)}
\]
Or, for better stability, the Log-Likelihood Ratio (LLR):

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

(3)

The MPA is an iterative algorithm based on the code’s graph for the computation of \( P(x=1|y) \), \( \ln(x_n) \) or \( \lambda_{x_n} \). The decoder operates as follow: i) Iteratively, the decoder evaluates the codeword’s check-constraints; ii) Notifies neighboring nodes of the confidence level of that bit being correct; iii) Each node, given the new incoming confidence level, recalculates his own confidence levels; iv) Again, 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, \( x_n \), of a given bit be 0 or 1 in the log-likelihood domain is given by:

\[ \begin{align*}
\lambda_{x_n} > 0 & \Rightarrow x_n = 1 \\
\lambda_{x_n} < 0 & \Rightarrow x_n = 0
\end{align*} \]

(4)

Thus the estimate that a received bit being 0 or 1 is depends only on the sign of \( \lambda_{x_n} \). For example, if \( y_n \) is received, \( \lambda_{x_n} = \ln(y_n) = \ln\left(\frac{P(x_n = 1|y_n)}{P(x_n = 0|y_n)}\right) \). 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 reliabilities are available: the reliability of \( x_n \) given \( y_n \), and the reliability of \( x_n \) given all the bits of \( y \) except \( y_n \). Taking into account these sources of reliabilities and Bayes’ theorem in both the numerator and denominator of (3):

\[ \lambda_{x_n} = \ln\left(\frac{P(x_n = 1|y_n)P(y_n|x_n = 1)}{P(x_n = 0|y_n)P(y_n|x_n = 0)}\right) = \ln\left(\frac{P(y_n|x_n = 1)}{P(y_n|x_n = 0)}\right) \times \ln\left(\frac{P(x_n = 1|y_n)}{P(x_n = 0|y_n)}\right) = \ln\left(\frac{P(y_n|x_n = 1)P(x_n = 1|y_n)}{P(y_n|x_n = 0)P(x_n = 0|y_n)}\right) \times \frac{P(x_n = 1|y_n)}{P(x_n = 0|y_n)} \]

(5)

The received channel’s sample \( y_n \) is independent of the set of received samples \( \{y_n\} \). Thus, (5) simplifies to:

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

(6)

Which, in turn, can be split in order to highlight the contributions of each \( x_n \) and \( y_n \) towards computing \( \lambda_{x_n} \):

\[ \lambda_{x_n} = \ln\left(\frac{P(y_n|x_n = 1)}{P(y_n|x_n = 0)}\right) + \ln\left(\frac{P(x_n = 1|y_n)}{P(x_n = 0|y_n)}\right) \]

(7)

The first term of (7), 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 \( n \).

Here 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:

From Fig. 4 we can see that at the noisy channel end, each bit measure, \( y_n \) plus associated noise, \( e_n \), is assigned a log-probability ratio, LLR, \( \lambda_{x,n,ext} \). 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, an iterative process called the 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) represents accurately 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 AWGN channel the intrinsic LLR for bit \( x_n \) is:

\[ \lambda_{x,n,intr}(y_n, \sigma^2) = \frac{2y_n}{\sigma^2} \]

(8)

Different channels types have different expressions for the LLRs which mirrors that particular channel noise model. As shown in [14], intrinsic LLR values that result from estimating the channel noise as greater than the real noise value, don’t affect the ECC but intrinsic LLRs computed over a optimistic noise parameter (lower than the real noise value) decrease the ECC ability.

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 cross-over, and, for the BPSK-AWGN channel, estimating the variance, \( \sigma^2 \). In a multichannel case, this method must be applied to each sub-channel 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:
\[
f(k, n, p) = \binom{n}{k} p^k (1 - p)^{n-k}
\]

where \( n \) is the number of trials, \( k \) successes (here "success" is the event of cross-over), and \( p \equiv p_s \), the estimated parameter.

For the BPSK with AWGN channels a solution is to use the Normal approximation to the binomial distribution [15], 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. Since the underlying probability distribution process is known (see (9)), methods such as Bayesian inference can be used to better and faster estimate the channel noise parameter. According to the Bayes theorem, naming the variables as hypothesis, data, and \( I \) (information), the data analysis and inference capability becomes [15]:

\[
P(\text{hypothesis} | \text{data}, I) = \frac{P(\text{data} | \text{hypothesis}, I) \times P(\text{hypothesis} | I)}{P(\text{data} | I)}
\]

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

\[
P(\text{new hypothesis}) = \frac{P(\text{previous hypothesis}) \times P(\text{new data})}{\text{normalization factor}}
\]

\textbf{A. LDPC ILE-Decoder}

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.

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{ILE-Decoder.png}
  \caption{ILE-Decoder inner block structure.}
  \end{figure}

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 \( y_i + e_i \). This LLR is then added (or composed) to the extrinsic LLR computed by the MPA in an iteratively fashion. After a preset number of iterations, a hard-decisioning is performed on the extrinsic LLR values yielding a decoded (and/or corrected) codeword \( \hat{y} \). 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 ECC performance.

\textbf{IV. SYSTEM PERFORMANCE}

The overall system architecture, encoders, decoder, modulator, demodulators, channels, but without the subchannel error estimation block, 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 multi-channel modulator (while being conveyed from serial to parallel ordering) where each subchannel can have different noise profiles (modulates bits represented by triangles). In the receiver the bits are demodulated and serialized into and estimated codeword \( \hat{y} \), which is sent to the LDPC decoder, yielding at the output an estimate for the original message \( u \).

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{Overall_System_Architecture.png}
  \caption{Overall System Architecture.}
  \end{figure}

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 II details the girth removal algorithm results. For each code (first column), its number of \( \{4, 6, 8\} \)-cycles are detailed.

\begin{table}[h]
  \centering
  \begin{tabular}{|c|c|c|c|}
    \hline
    Code & Number of n-cycles & 4 & 6 & 8 \\
    \hline
    \( H_1 \) & 276 & 1252 & 984 \\
    \hline
    \( H_2 \) & 10 & 2098 & 56 \\
    \hline
  \end{tabular}
  \caption{Listing of each code n-cycles before (\( H_1 \)) and after (\( H_2 \)) the removal of small cycles}
  \end{table}

It’s easily noticeable from Table II that there are much less 4-cycles 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 cycles.

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 \( H_1 \).

Just displaying one subcarrier for simplicity, Fig. 8 from (a) to (i) details the iterative unfolding of an example estimation process. The estimation process starts with “full ignorance”, the estimated parameter could be uniformly distributed in the whole 0 to 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. By the last iteration, Fig. 8(i), the maximum of the pdf is already located near the real \( p_s = 0.30 \).
This paper presented methods for improving an LDPC coder and decoder through changes in its structure. Small to medium LDPC codes, which tend to perform good enough while keeping the system complexity low, were targeted in this paper. 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 system. A fast process was detailed for the encoder and an improved version of a decoder – the ILE-Decoder – was derived. The ILE-Decoder resulted from the addition of an intrinsic LLR estimator to the LDPC decoder, in order to better estimate each subchannel noise parameters.

Simulation results for the new coder/decoder were presented, which confirm its better performance regarding the classic structures.

The PLC was chosen as system environment but the results can apply to other medium requiring small to medium LDPC codes.

REFERENCES