Simulador de Sistema PLC Utilizando OFDM Adaptativo com Códigos LDPC

José Manuel Gouveia de Miranda Ribeiro Vaz
November 5, 2010

Abstract

Despite LDPC codes performing admirably for large block sizes - being mostly resilient to low levels of channel SNR and errors in channel equalization - small and medium sized codes tend to be affected by these two factors. For these small to medium codes, a new method is presented that helps to reduce the dependency of correct channel equalization, without changing the inner workings or architecture of existing LDPC decoders. This goal is achieved by using LDPC only decoder-side information - gathered during standard LDPC decoding - that is used to improve the estimation of channel parameters, thus improving the reliability of the error code correction, while reducing the number of required iterations for a successful decoding. This scheme is suitable for application to multicarrier communications, such as OFDM. A PLC-like environment was chosen as the basis for presenting the results.

1 Introduction

The design and project 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 the transmitted message is corrupted. The addition of any sort of FEC always increases overall system complexity and reduces usable bandwidth, but at the same time, it lowers transmission error rates.

Several modulation schemes can be used to transmit data over the Power Line Communication (PLC) channel, but Orthogonal Frequency-Division Multiplexing (OFDM) is well suited to lessen most of the channel's conditions such as interference, attenuation and fading. Adding the ability of FEC to an OFDM system gives rise to what is usually called Coded Orthogonal Frequency-Division Multiplexing (COFDM). This document deals with improvements made to an OFDM system when using a type of FEC called Low-Density Parity-Check (LDPC) coding.

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. They were firstly proposed in the 1960's by Gallager [1], in his doctoral dissertation. However, for the most part of the next 30 years, those codes remained as forgotten. In the mid 90's these codes were “rediscovered” independently by different researchers [2], who noticed the very desirable properties of linear block codes with Low-Density (i.e. sparse) Parity-Check matrices. The current maximum error correction capacity for LDPC codes currently falls within 0.0045dB of the Shannon limit [3], although the feasibility of such codes (and associate decoders) for real-time applications can be questioned due to the excessive requirements of computational resources.

2 Channels

In evaluating the response of a given system to noise/perturbations, it is possible to use different types of noisy channel, where each type of model usually highlights some specific kind of
real-world perturbation.

For communication systems that employ LDPC codes, the following three channels tend to be used in modeling the various kinds of errors (figure 1) a communication system is subjected to:

(a) Model and parameters of BSC channel
(b) Model and parameters of BEC channel
(c) Model and parameters of BPSK-AWGN channel

Figure 1: Channels used for modeling frequency-selective noise

The Binary Symmetric Channel, a channel that models the kinds of noise that appear as bit flips. The BSC is a channel where a sent bit has a probability $p_c$ (probability of cross-over) of being flipped, and a $1-p_c$ probability of being transmitted without being affected by noise.

The Binary Erasure Channel (BEC) models the types of noise that erase transmitted bits—that is, the receiver is unable to reach a conclusion whether the original bit was either a 1 or 0. The probability of a given bit being erased is $p_e$, while the probability of a bit being transmitted without being erased is $1-p_e$.

The last channel (1c) is a Binary Phase-Shift keying (BPSK) modulated channel with Additive White Gaussian Noise (AWGN). This channel models the case of AWGN, one of the most common cases of “real-world” noise, where the original bits are disturbed by an amount of noise proportional to $\sigma^2$.

Of these three, only the BSC and the BPSK-AWGN channels, will be amply used in this thesis.

As previously stated, it is assumed that the proposed communication system operates in a frequency-selective noise environment, typical of PLC or Digital Subscriber Line (DSL) environments. The best way to deal with frequency-selective noise, is to discard all sub-bands affected by noise, while using the remainder sub-bands. By design OFDM systems already split modulated data through several subcarriers / sub-bands. In this thesis, it is introduced 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 - figure 2. Each subchannel 1, 2, ..., N, has its own probability of cross over, $p_{c1}, p_{c2}, \ldots, p_{cN}$.

In this stacked model, each single BSC is intended to model a particular frequency of frequency-selective noise.

Figure 2: Frequency-selective noise BSC bank channel

Here lies the major assumption in this work: for the modifications made to the LDPC decoder, it is irrelevant where in the electromagnetic spectrum the first and subsequent subcarriers are located [Hz], how each one is spaced, interleaved, what was the modulation type, etc. The additions and modifications done to the LDPC decoder result that 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 possi-
ble the work is this thesis is applicable.

3 Low-Density Parity-Check Coding

The parameters (and its symbols) that define a LDPC code are introduced in Table 1, which provides a convenient reference for all LDPC used symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>Parity-Check Matrix (PCM)</td>
</tr>
<tr>
<td>$u$</td>
<td>source message vector - user message/user bits</td>
</tr>
<tr>
<td>$x$</td>
<td>transmitted codeword vector - encoded sequence of user bits</td>
</tr>
<tr>
<td>$y$</td>
<td>received codeword vector: original codeword $x$ after being sent through a noisy channel</td>
</tr>
<tr>
<td>$G$</td>
<td>Generator matrix</td>
</tr>
<tr>
<td>$g$</td>
<td>girth - length of $G$ shortest cycle of $H$</td>
</tr>
<tr>
<td>$j$</td>
<td>$H$ column weight - number of $H$ non-zeros per column</td>
</tr>
<tr>
<td>$k$</td>
<td>$H$ row weight - number of $H$ non-zeros per row</td>
</tr>
<tr>
<td>$N$</td>
<td>number of $H$ columns, variable nodes and codeword length</td>
</tr>
<tr>
<td>$M$</td>
<td>number of $H$ rows and number of check nodes</td>
</tr>
<tr>
<td>$K$</td>
<td>source message size</td>
</tr>
<tr>
<td>$R$</td>
<td>code rate</td>
</tr>
<tr>
<td>$c_m$</td>
<td>check-node $m$</td>
</tr>
<tr>
<td>$v_n$</td>
<td>variable-node $n$</td>
</tr>
</tbody>
</table>

Table 1: Notation for LDPC codes used throughout this document

3.1 LDPC codes defined

As a linear-block code, an LDPC code has a parity-check matrix representation, denoted by $H$, although to be considered as a LDPC code, it has to satisfy some specific construction rules and parameters. All LDPC code parameters are linked to several properties of the $(M \times N)$ $H$ matrix. The LDPC parameters are: $(N, M, j, k)$, where $N$ is code block-length, $M$ is the number of constraints - or check nodes - $j$ and $k$ are respectively $H$ column and row weight (weight: the number of non-zero elements in a row/column). The codeword size, $N$, is related to the message size $K$, by the expression $N = M + K$. The code rate (the portion that is used for message data) is given by $R = (N - M)/N = 1 - M/N$.

A code is said sparse or low-density (hence its name) when $k \ll N$ and $j \ll M$. This is the first main “feature” of LDPC codes: the $H$ matrix sparseness/low-density.

An example of an $(N = 8, M = 4, j = 2, k = 4)$ LDPC code (non-sparse for demonstration only purposes), is the following $4 \times 8$ matrix:

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$ (1)

The rate for this code is $R = 1 - 4/8 = 0.5$, which is a code rate of 50%. Thus for every 4 message bits another 4 bits are added as redundancy.

3.2 LDPC encoding

Because LDPC codes are block codes, the same encoding methods apply. The textbook approach for encoding linear block codes is a standard vector-matrix operation, $x = uG$. The generator matrix usually has the form $G = [I_k|P]$, where $P$ is a dense matrix containing the parity information, obtained by reducing the original $H$ matrix to the systematic form $H = [-P^T|I_{n-k}]$.
through Gaussian elimination. Figure 3b is graphical representation of an example parity check matrix in systematic form, where black pixels represent non-zero elements. The original H in sparse representation is depicted in 3a.

### 3.3 Iterative LDPC decoding

One of the major challenges in implementing LDPC coding system is the complexity or feasibility of the decoder. Although LDPC decoding is a NP problem, Gallager also introduced a near-optimal decoding [4] algorithm, the Message-Passing Algorithm (MPA).

The purpose of iterative decoding algorithms applied to the noisy channel problem is to compute and estimate of original codeword x given the sampled channel output y, while minimizing the decoded sequence’s bit error rate, ̂x. This is achieved by computing the Maximum a posteriori probability (MAP) for \( P(x|y) \).

For LDPC decoding, the most efficient known method for computing each of the \( x_n \) bits is the a posteriori probability (APP), \( P(x_n = 1|y) \). Because the decoding is performed on graphs, the particular algorithm used for computing the APP is the Message-Passing Algorithm (MPA). This method iteratively computes the probability of a transmitted codeword bit \( (x_n) \) being 1, given all received codeword’s bits \( y = [y_0 \ y_1 \ldots y_{N-1}] \), that is \( P(x_n = 1|y) \).

Instead of having to deal separately with the \( P(x_n = 1|y) \) or \( P(x_n = 0|y) \), the APP ratio can be used:

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

Or, for better numerical stability, the log APP ratio, Log-Likelihood Ratio (LLR), can be used:

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

The MPA is an iterative algorithm based on the code’s graph for the computation of \( P(x = 1|y) \), \( l(x_n) \) 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, recomputes his own confidence levels, and again iv 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 of a given bit be 0 or 1 in the log-likelihood domain is given by:

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

Thus the estimate that a received bit being 0 or 1 is depends only on the sign of \( \lambda_{x_n} \), \( \hat{x}_n = \text{sign}(\lambda_{x_n}) \). Establishing the value of \( \hat{x}_n \) only using the sign information is quite a coarse evaluation. The magnitude of the \( \lambda_{x_n} \) carries the reliability of the estimate of \( \hat{x}_n \) being towards 0 or 1. For example, for \( \lambda_{x_n} = 0 \), \( \hat{x}_n \) has the same likeness of being a 0 or 1.

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 \( y_n \), that is: \( y \setminus y_n \). Taking into account these two sources of reliabilities, and applying Bayes’ theorem to both the numerator and denominator of equation 3.

\[
\begin{align*}
\lambda_{x_n} &= \ln \left( \frac{P(x_n = 1|y_n, y \setminus y_n)}{P(x_n = 0|y_n, y \setminus y_n)} \right) = \\
&= \ln \left( \frac{P(y_n|x_n = 1, y \setminus y_n)}{P(y_n|x_n = 0, y \setminus y_n)} \cdot \frac{P(x_n = 1|y \setminus y_n)}{P(x_n = 0|y \setminus y_n)} \right) \\
&= \ln \left( \frac{P(y_n|x_n = 1, y \setminus y_n)}{P(y_n|x_n = 0, y \setminus y_n)} \right)
\end{align*}
\]  

The received channel’s sample \( y_n \) is independent of the set of received samples \( y \setminus y_n \). Thus, the previous expression simplifies to:

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

4
Which in turn can be split in order to highlight the contributions of each \( x_n \) and \( y_n \) towards computing \( \lambda_{x_n} \):

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

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 the channel samples except the current bit \( n \).

### 4 Building LDPC codes

Several parameters dictate 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 an efficient runtime encoding and decoding methods, else it isn’t suitable for real-time and/or low-latency usage.

The ultimate benchmark for a LDPC code usually is its ECC ability. Other parameters and properties can also affect a code performance. These, girth size, and code structure. These parameters are detailed in the following subsections.

#### 4.1 Girth and small-cycles removal

A code’s minimum girth is also another parameter affecting performance. The girth is defined as the smallest loop/cycle (the path from a given node back to itself) within a code’s graph representation.

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 fewer iterations, and fewer nodes are reached. Thus, smaller cycles biases incoming messages of all its spanning nodes towards the reliability updates of the cycles members.

![Figure 4: Thick solid: A 4-loop defined by vertexes \( v_3, c_1, v_7, c_3 \), in both tanner graph (left) and parity-check matrix forms (right). Dashed: a 6-loop defined by vertexes \( v_3, c_1, v_5, c_2, v_1, c_3 \).](image)

Smaller cycles, (see figure 4) mean that less reliability for a given bit can be gathered by the decoder. Consider the (dashed) small loop (4-cycle) depicted in the figure, spanning the vertexes \( c_1, v_3, c_3 \) and \( v_7 \): At variable node \( v_3 \), there can be only reliability updates coming from \( c_1 \) and \( c_3 \). After 2 iterations (two “up” and two “down” half-iterations), \( v_3 \) will receive updated reliabilities weighting the whole \( c_1v_3c_3v_7 \) cycle. There are of course larger cycles that also have \( v_3 \) as a vertex (see the dashed 6-cycle \( c_1v_3v_5c_2v_1c_3 \)). But the consequence is that every 2 iterations \( v_3 \) will receive through \( c_1/c_3 \) similar updates, weighting mostly \( v_1v_3v_5c_2 \). 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 \( v_3 \) will be heavily biased towards the information updates that happen over the small 4-cycle.

This is a limiting result, because the condition of convergence of the MPA assumes that the cycles spanning at each node are infinite.

Most authors [?] acknowledge the desirab-
ity of removing small loops in LDPC graphs, due
to the loss of error coding performance caused
by such small cycles. The typical algorithm for
generating LDPC codes usually makes a mini-
imal 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 po-
sitions (thus forming a rectangle of four 1s in
the matrix - figure [4]. With some shuffling and
row/column swapping it’s usually possible to re-
move a 4-cycle. Longer cycles, aren’t as easily
removed as 4-loops. Approaches such as given
in [? ] which just delete small cycles, tend to
deeply change the properties of a LDPC code.

5 Conclusions

The proposed methods help retrieve some ex-
tra performance from LDPC decoded. Although
not suited for all decoder implementations, these
methods are mostly orientates for environments
where retransmission are not possible or expen-
sive.

References


based on very sparse matrices,” Cryptogra-
phy and Coding 5th IMA Conference, no.

[3] S.-Y. Chung, G. D. Forney, T. J. Richardson,
and R. Urbanke, “On the Design of Low-
Density Parity-Check Codes within 0.0045
dB of the Shannon Limit,” IEEE Communi-

Codes,” IRE Transactions on Information
Theory, vol. 8, no. 1, pp. 21–28, January
1962.