Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 54(68), Fascicola 1, 2009

Decoding Performance of Turbo-Codes and LDPC-Codes with Short Blocklength

Wolfgang Proß, Franz Quint

Abstract – In this paper the decoding performance of a Turbo-Code and a LDPC-Code are compared. Both exhibit a binary blocklength of 504 and a coderate of 0.5. After an explanation of the several channel code's construction methods the results of the simulations are depicted in terms of the Bit Error Rate (BER). For a channel model the Additive White Gaussian Noise Channel (AWGNC) has been used. The decoding was done with the iterative Belief Propagation (BP)-decoding algorithm. If carefully constructed, the LDPC-Code is clearly favorable over the applied Turbo-Code.

Keywords: Turbo-Code, LDPC-Code, GF(q), PEG

I. INTRODUCTION

Channel coding is very important in digital systems and effectively improves the efficiency concerning transmission of binary information in the presence of interferences. Since the introduction of A mathematical theory of communication [9] by Claude E. Shannon in 1948 channel coding schemes have tried to achieve the Shannon limit. It is defined as a lower bound on the Signal to noise ratio (SNR) at which an appropriate coding scheme can barely allow for a transmission. When Turbo-Codes were presented by Berrou, Glavieux and Thitimajshima [2] in 1993 the concatenated coding scheme was proved to come up very close to the Shannon limit. Low-Density Parity-Check (LDPC) – Codes were already published in 1962 by Robert Gallager [5] and showed an asymptotically optimal decoding performance. At first LDPC-Codes were forgotten because of computational burden but since computation power has experienced a high increase LDPC-Codes became interesting again. Since MacKay and Neal had rediscovered them in 1995 [7] various authors have published improvements concerning the construction methods of LDPC-Codes. This yielded in a better decoding performance where LDPC-Codes have become competitive to Turbo-Codes even for short blocklength.

II. TURBO-CODES

A Turbo-Code is attained by a serial or parallel concatenation of several channel-codes. Furthermore the appropriate decoder processes soft decision values. A hard decision of a received bit can take the values zero or one with respect to the estimated sent bit. In contrast soft decisions carry the information of the probability of a bit to be a zero or a one. To minimize the required computing power soft decision values are often processed in the Log-domain. A convenient format to do so is the Log-Likelihood-Ratio (LLR). The Turbo-decoder takes advantage of the more precise soft decision values and constantly refines the estimations of each bit in an iterative process. This is done by an exchange of extrinsic data among the several component decoders. Fig.1 shows the functional principle of a Turbo-Code. One can see the parallel concatenated Turbo-encoder on the left side of the channel and the appropriate Turbo-decoder on the right, consisting of two component encoders and decoders respectively. The first component encoder E1 encodes the information word. E2 gets an interleaved version of the systematic part as an input signal. The feedback loops in the Turbo-decoder that carry the extrinsic data are depicted by the dashed grey lines. At the end of the decoding procedure a hard decision is performed.

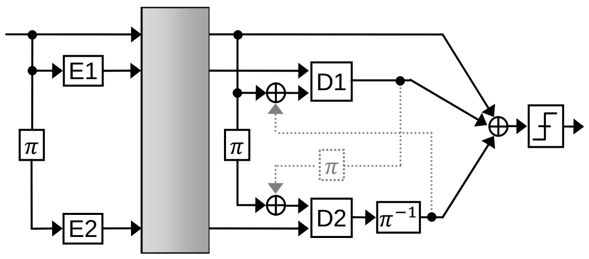

Fig. 1. Principle of a Turbo-Code

The similarity of the Turbo-Decoder's functional principle to a Turbo charger gives rise to the name

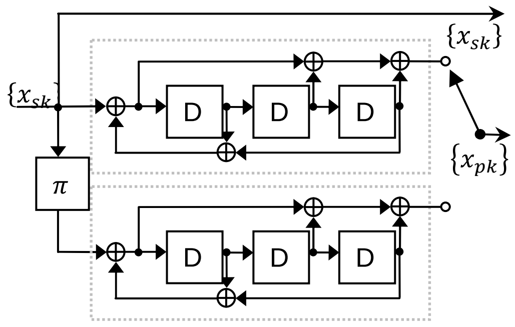

Fig. 2. Turbo-encoder with r=1/2

Turbo-Code.

A. *Turbo-encoder*

The Turbo-encoder that is used here consists of two parallel concatenated 8-state convolutional encoders described by the generator polynomial in octal notation:

$$G = \frac{p}{q} = \frac{13}{15} \qquad (1)$$

The Turbo-encoder is shown in Fig. 2.
The systematic part is denoted as x_{sk} with $1 \leq k \leq n/2$. Encoder 1 encodes x_{sk} and outputs x_{p_1k} (parity part one). The second component encoder gets the interleaved version of the information word $\pi(x_{sk})$ and outputs x_{p_2k}. By use of an appropriate puncturing pattern x_{p_1k} and x_{p_2k} get compacted into x_{pk}. A code word of a ½-rate Turbo-Code then comprises of the systematic part and the packed parity part so that

$$x_{sp} = \{x_{sk}\ x_{pk}\}.$$

B. *Turbo-decoder*

The Turbo-decoder shown in Fig. 3 processes Log-Likelihood-Ratios (LLR's) computed from the output of the established channel.

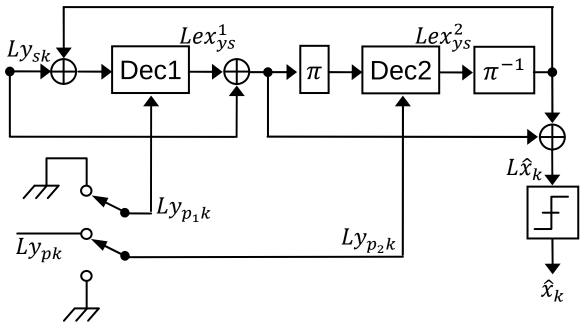

Fig. 3. Turbo-decoder with r=1/2

In this case an Additive White Gaussian Noise Channel (AWGNC) is modeled. Thereby white Gaussian noise is added to the BPSK-modulated codeword depending on the signal to noise ratio (SNR). In channel coding simulations the SNR is usually defined as E_b (energy per bit) divided by N_0 (spectral noise density). With $x_i \in \{1, -1\}$ being a sent bit, the conditional probability of a bit y_i received by the decoder is then distributed as follows:

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i \pm x_i)^2}{2\sigma^2}}, 1 \leq i \leq n \qquad (2)$$

The LLR for the AWGNC is then:

$$L(y_i|x_i) = ln\frac{p(y_i|x_i=+1)}{p(y_i|x_i=-1)} = \frac{2}{\sigma^2} \cdot y_i. \qquad (1)$$

The two component decoders DEC1 and DEC2 in Fig. 3 are BCJR-decoders named after Bahl, Cocke, Jelinek, and Raviv that presented a trellis-based decoding method in 1974 in [1]. Thereby the *a posteriori* probability $p(x_i = b|y_i)$ of a BPSK-modulated sent bit $b \in \{1, -1\}$ is maximized. Ly_{sk} represents the LLR's of the received systematic part and Ly_{pk} the LLR's of the compressed parity parts. For each bit of the parity parts that was punctured in the encoding procedure a zero is inserted instead and the LLR's of y_{p_1k} and y_{p_2k} get computed. The extrinsic data of the first component decoder Dec1 is then obtained by use of the systematic part Ly_{sk}, the extrinsic part of the second decoder Lex^2_{ys} and the parity part 1 Ly_{p_1k}. When the desired number of iterations have been processed, the extrinsic data of both component decoders and the sytematic part get added and a hard decision is performed. The result represents the estimation of the original information word.

C. *Turbo-Code simulations*

For the following simulations the allzero codeword is used which is a codeword that exhibits n zeros. The allzero codeword is always a valid codeword for a linear code. To depict the error correcting capabilities of the code, the bit error ratio (BER) is plotted on the y-axis of the graph (Fig. 4) while the according E_b/N_0-values are plotted on the x-axis. The BER is obtained by dividing all errors occurring in a decoded codeword by the length of the code word n where $n = 504$. The simulation as well as the appropriate Turbo-decoder were implemented in Matlab.

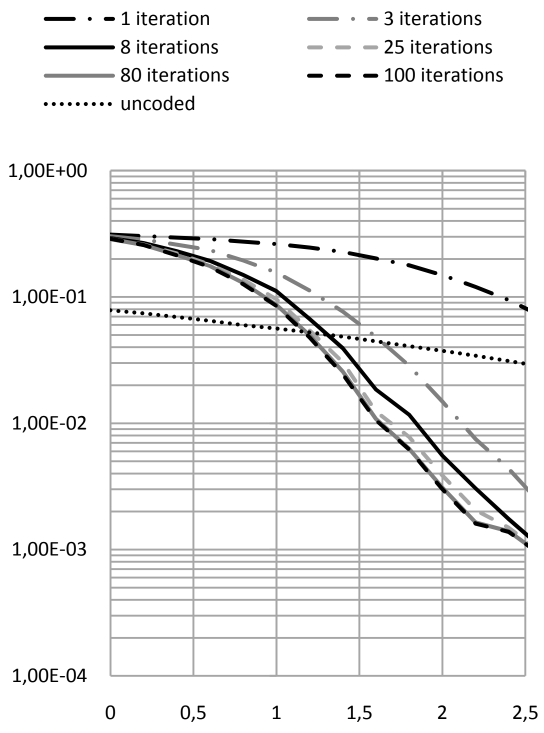

Fig. 4. BER of Turbo-Code , r=1/2, n=504

III. LDPC-CODES

To create a Low-Density Parity-Check (LDPC) –Code one constructs the underlying Parity-Check-Matrix (PCM) or alternatively a Tanner-graph. The PCM and the Tanner-graph of a LDPC-Code are interchangeable and fully represent the code. In Fig. 5 the PCM of a LDPC-Code is shown. The name of LDPC-Codes stems from the fact that the PCM is always sparse and thus possesses a low-density of nonzero elements. Furthermore the rows of the PCM represent parity-check equations that can be seen at the bottom of Fig. 5.

$$H_{m\times n} = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ \vdots & \ddots & \vdots \\ h_{m1} & \cdots & h_{mn} \end{bmatrix} =$$

$$\begin{array}{c} \begin{bmatrix} \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \\ \begin{array}{ccccccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 & s_9 \end{array} \end{array}$$

$$\begin{aligned} c_1 &\rightarrow s_1 + s_2 + s_3 = 0 \\ c_2 &\rightarrow s_4 + s_5 + s_6 = 0 \\ c_3 &\rightarrow s_7 + s_8 + s_9 = 0 \\ c_4 &\rightarrow s_2 + s_5 + s_7 = 0 \\ c_5 &\rightarrow s_1 + s_4 + s_6 = 0 \\ c_6 &\rightarrow s_2 + s_8 + s_9 = 0 \end{aligned}$$

Fig. 5. PCM of an LDPC-Code, r=1/3, n=9

A PCM always possesses n columns and $m = n - k$ rows where n is the codeword's blocklength and k stands for the length of the information word. The coderate is then $r = k/n$. With the help of the Gaussian elimination any PCM can be transformed to

$$H^{m\times n} = [P^{T\,m\times k}\; I^{m\times m}] \tag{4}$$

with I being the identity matrix. From this the generator matrix

$$G^{k\times n} = [I^{k\times k} \quad P^{k\times m}] \tag{5}$$

is derived. A codeword is then obtained by multiplying the information word x_{sk} with the generator-matrix $G^{k\times n}$. In Fig. 6 the corresponding Tanner-graph to the PCM in Fig. 5 is shown. This bipartite graph comprises of n symbol-nodes and m check-nodes representing the columns and rows of the PCM respectively. These nodes are connected via edges corresponding to the entries in the PCM. The black edges adjacent to s_1 in Fig. 6 for example correspond to the bold nonzero entries in Fig. 5.

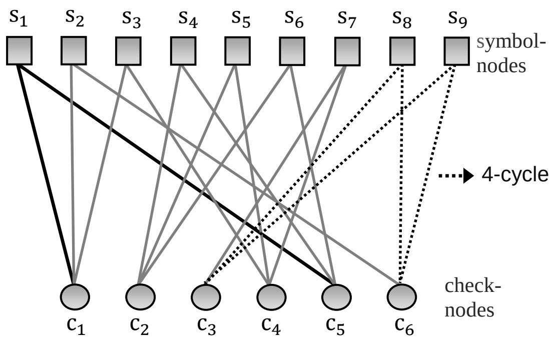

Fig. 6. Tanner-graph corresponding to the PCM in Fig. 5

The decoding performance of a LDPC-Code is highly dependent on the cycles that code exhibit. A cycle is a closed path of consecutive edges that connect a node with itself. The number of involveld edges defines the length of a cycle. In Fig. 6 a 4-cycle is shown. For each symbol-node s_i in a Tanner-graph the length of the shortest cycle passing through this symbol-node is denoted as local girth g_{s_i}. Global girth g is defined by the length of the shortest cycle that exists in a Tanner-graph and so

$$g = \min_i\{g_{s_i}\}. \tag{6}$$

A low global girth has a harmful impact on the decoding performance which is thus mainly dependent on the construction of the PCM or the Tanner graph. This is the reason for optimizing the construction method in reference to the resulting decoding performance.

A. *Regular & Irregular*

In [5] Gallager introduced LDPC-Codes and proposed a pseudo-random construction method for the PCM of a *regular* LDPC-Code. The matrix of a regular LDPC-Code always possesses exacly γ nonzero elements in each column and ρ in each row and thus all check-nodes and symbol-nodes share the same number of adjacent edges respectively. The LDPC-Code shown in Fig. 5 and Fig. 6 represents a so called Gallager-code described by $(n, \rho, \gamma) = (9,3,2)$. In contrast to regular LDPC codes, irregular codes exhibit several row and column weights. They are described through the use of the symbol-node degree distribution

$$\Lambda(x) = \sum_{i\geq 2}^{d_s^{max}} \Lambda_i \cdot x^i, \tag{2}$$

where d_s^{max} is the maximum number of edges connected to a symbol-node in the graph and Λ_i is the fraction of symbol-nodes connected to i check-nodes. Since it is a distribution it follows:

$$\sum_{i\geq 2}^{d_s^{max}} \Lambda_i = 1. \tag{3}$$

B. *Progressive-Edge-Growth*

The *Progressive-Edge-Growth (PEG)* algorithm introduced by Hu, Eleftheriou and Arnold in [6] constructs the Tanner-graph of an LDPC-Code by means of progressivley establishing edges between

the symbol- and check-nodes. Each time an $edge\,(s_i, c_j)$ is placed the local girth g_{s_i} of the involved symbol-node s_i is maximized. By constantly maximizing the local girth the global girth is maximized as well because of (6).
There are three different situations when choosing a check-node c_j in order to establish an edge (s_i, c_j):

1. If it is the first edge to get connected to a symbol-node s_i → choose the check-node having the lowest check-node degree (fewest connected edges) under the current graph settings.
2. If there are still check-nodes that are not already connected to the current graph → choose one of them.
3. If neither of the two former cases are true → establish a PEG-tree with s_i as a root of that tree. Then choose a check-node of the bottom-layer.

If a third edge should be connected to symbol-node s_1 in Fig. 6 for example, a PEG-tree has to be created in order to find an appropriate check-node c_j. The root of that PEG-tree is s_1. In Fig.7 the creation process of the PEG-tree is depicted.

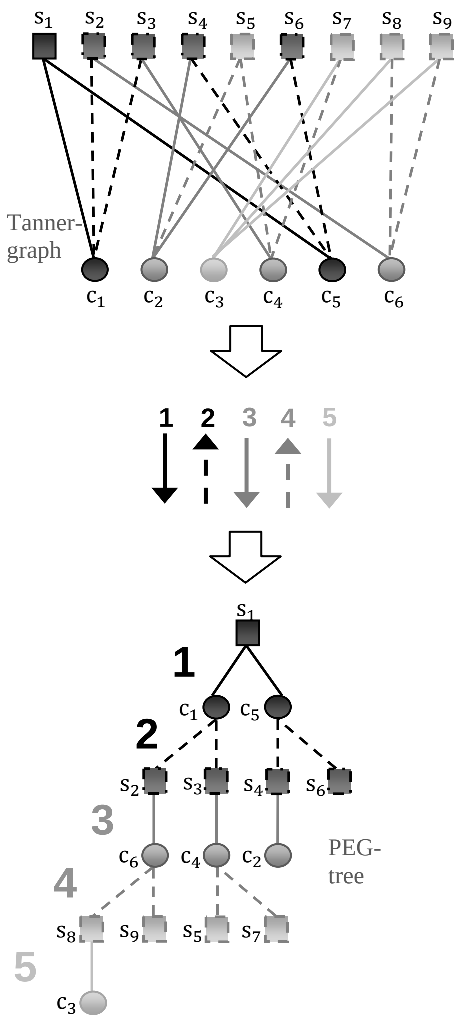

Fig. 7. Constructing a PEG-tree to find an appropriate check-node

In correspondence to the third case above, the edge to be established would be $edge(s_1, c_3)$. Hu, Eleftheriou and Arnold also propose to enhance the matrix construction by a partitioning of the PCM that becomes to:

$$H_{m\times n} = \left[H^p_{m\times m}, H^d_{m\times(n-m)}\right], \quad \text{with}$$

$$H^p = \begin{bmatrix} 1 & h^p_{12} & \cdots & h^p_{1m} \\ 0 & 1 & \cdots & \vdots \\ \vdots & \cdots & \ddots & h^p_{(m-1)m} \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{m\times m} . \qquad (4)$$

By doing so the encoding time of the resulting *linear time encodable* LDPC-Code increases with n instead of with n^2 as it usually does. The codeword's parity bits can then be calculated according to:

$$p_i = \sum_{j=i+1}^{m} h^p_{ij}\, p_j + \sum_{j=1}^{n-m} h^d_{ij}\, d_j \pmod 2 \qquad (5)$$
where $i = m, m-1, \ldots, 2, 1$.

C. *LDPC-Codes defined over GF(q)*

By an increase of a binary PCM's column weight, the Hamming weight spectrum and hence the decoding performance is improved. The drawback is that if the PCM possesses more nonzero entries, the number of cycles increases which results in a degradation of the codes error correction capabilities. When moving to GF(q) the mean column weight increases while the number of cycles in the nonbinary Tanner graph remains the same [3]. The construction methods to attain a nonbinary PCM do not differ from those of binary LDPC codes. In contrast to the latter, the PCM of nonbinary LDPC codes possesses elements defined over the Galois field $GF(q) = GF(2^p)$. Thereby the nonzero entries in H are generated through the use of a primitive polynomial $p(z)$ where $p(z) = 0$. It is also essential to realize calculations required during the decoding process in the Galois field $GF(q)$. They are based on a polynomial representation of the elements. A $GF(q)$ symbol is represented by p bit, whereas the exponents of the corresponding polynomial stand for the indices of the several bits and the coefficients for their value.

D. *LDPC-Code simulations*

As in the case of the Turbo-Code simulations the AWGNC and the allzero codeword with binary codeword length of 504 were applied in all LDPC-Code simulations.
The simulation results in Fig. 8 were attained by use of a irregular linear time encodable LDPC-Code of rate $r = 1/2$. The PCM was constructed based on the symbol-node degree distribution $\Lambda(x) = 0{,}47532x^2 + 0{,}279537x^3 + 0{,}0348672x^4 + 0{,}108891x^5 + 0{,}101385x^{15}$ that is taken from [6]. For the decoding the log-domain based Belief-

Propagation (BP) decoder described in [8] has been applied.

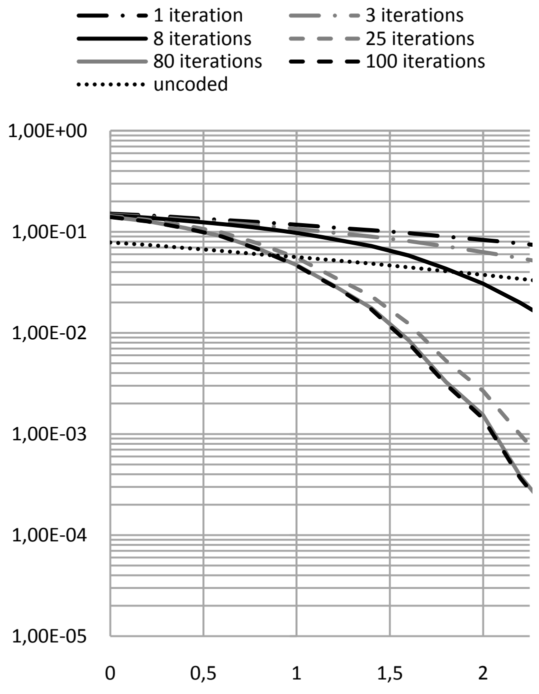

Fig. 8. BER of irregular linear time encodable LDPC-Code, n=504, r=1/2

The following simulation results in Fig. 9 show the BER of an irregular linear time encodable PEG LDPC-Code over GF(64) with coderate $r = 1/2$. The symbol-node degree $\Lambda(x) = 0{,}94x^2 + 0{,}05x^3 + 0{,}01x^4$ for the construction of the PCM has been chosen in correspondence to [6]. The decoding was done with a FFT-based BP-decoder as described in [4].

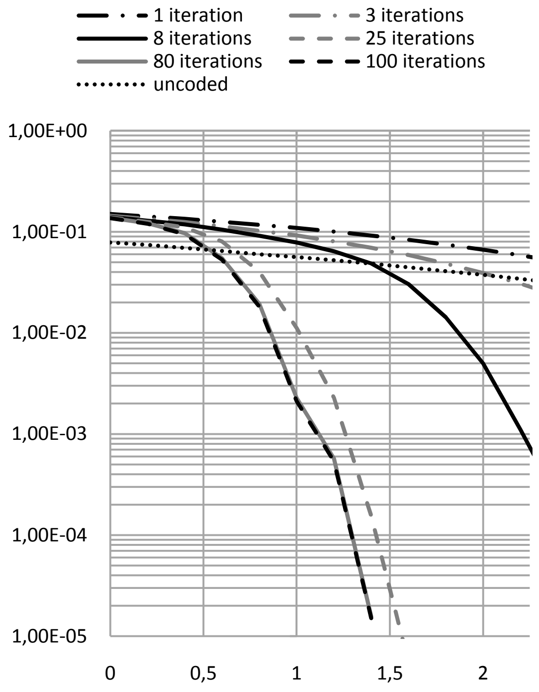

Fig. 9. BER of irregular linear time encodable LDPC-Code over GF(64), n=84, r=1/2

IV. TURBO-CODE VS. LDPC-CODE

In Fig. 10 the most promising of the above LDPC-Codes (the one defined over GF(64)) and the Turbo-Code explained above are compared in terms of BER. A codeword of the LDPC-Code is of symbol-length $n = 84$. Because $GF(64) = GF(2^6)$ a symbol consists of 6 bit so that the binary length becomes to $6 \cdot 84 = 504$ and thus is comparable to the Turbo-Code.

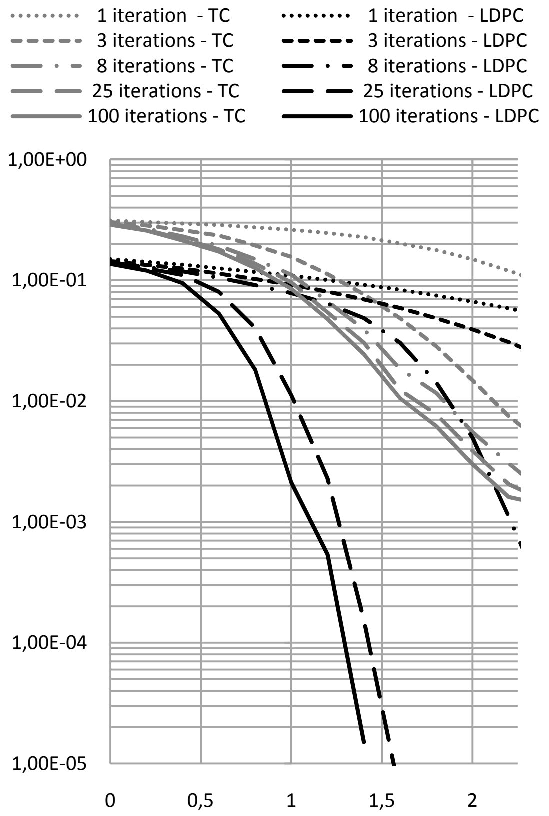

Fig. 10. BER comparison between irregular linear time encodable LDPC-Code over GF(64) and a Turbo-Code, both with binary codeword-length of 504 and coderate 0.5

As seen in Fig. 10 the Turbo-Code decreases the BER between 1 and 3 iterations much more than the LDPC-Code. But that changes for more iterations where the Turbo-Code improves his estimation only marginal between 25 and 100 iterations whereas the LDPC-Code highly increases the coding-gain. For 25 or more iterations the LDPC-Code is clearly favorable over the Turbo-Code in terms of BER.

V. CONCLUSION

The PEG algorithm offers an effective construction method for high girth LDPC codes that are competitive to Turbo codes. Especially when moving to higher order Galois fields GF(q) irregular PEG LDPC codes beat the applied Turbo code even for a short code word length. As a result of this comparison we construct a near Shannon limit coding scheme for

2D-Data Matrix code applications using the explored nonbinary linear-time encoding PEG LDPC code. This leads to better results in terms of BER and computational burden.

VI. ACKNOWLEDGMENT

This work is part of the project MERSES and has been supported by the European Union through its European regional development fund (ERDF) and by the German state Baden-Württemberg.

REFERENCES

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)", *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284-287.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, Eds., *Near Shannon limit error-correcting coding and decoding: Turbo-codes*, 1993.

[3] M.C. Davey, "Error-correction using low-density parity-check codes", *Univ. of Cambridge PhD dissertation.*

[4] D. Declercq, and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF (q)", *IEEE Transactions on Communications*, vol. 55, no. 4, p. 633.

[5] R. Gallager, "Low-density parity-check codes", *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21-28.

[6] X.Y. Hu, E. Eleftheriou, and D.M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs", *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386-398.

[7] MacKay D.J., and Neal R.M., *Good Codes Based on Very Sparse Matrices: Cryptography and Coding. 5th IMA Conf.(Cirencester, UK), LNCS 1025*: Berlin: Springer, 1995.

[8] W.E. Ryan, "An Introduction to LDPC Codes", The University of Arizona, 2003.

[9] C.E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. 3, p. 4.