Log-domain decoding of LDPC codes over $GF(q)$

Henk Wymeersch, Heidi Steendam and Marc Moeneclaey
DIGCOM research group, TELIN Dept., Ghent University
Sint-Pietersnieuwstraat 41, 9000 GENT, BELGIUM
E-mail: {hwymeers,hs,mm}@telin.ugent.be
Tel.: +32-9-264.89.00
Fax.: +32-9-264.42.95

***Abstract*— This paper introduces a log-domain decoding scheme for LDPC codes over $GF(q)$. While this scheme is mathematically equivalent to the conventional sum-product decoder, log-domain decoding has advantages in terms of implementation, computational complexity and numerical stability. Further, a suboptimal variant of the log-domain decoding algorithm is proposed, yielding a lower computational complexity. The proposed algorithms and the sum-product algorithm are compared both in terms of simulated BER performance and computational complexity.**

I. Introduction

Regular low-density parity-check (LDPC) codes were introduced by Gallager in [1], along with several message passing decoding algorithms. It has recently been shown that regular binary LDPC codes, when decoded using the sum-product algorithm (SPA), approach the capacity of the AWGN channel [2]. Conventional LDPC codes were extended in two ways. On the one hand, by removing certain constraints on the parity check matrix, so-called irregular LDPC codes were shown to outperform regular LDPC codes [3]. On the other hand, by allowing coding and decoding to take place in higher order Galois fields, better codes could be constructed [4]. Such codes also have superior performance in the presence of burst errors [5]. In [6], it was shown that the best performing LDPC codes (in terms of bit error rate (BER) vs. signal-to-noise ratio (SNR)) are irregular non-binary LDPC codes.

The direct implementation of the decoding algorithm for binary codes in the probability domain (i.e., the SPA) has several drawbacks as compared to an implementation, denoted here by log-SPA, based on log-likelihood ratios (LLR): the direct implementation is more sensitive to quantization effects and requires more quantization levels than when using LLRs [7], [8]. The SPA requires message multiplications, whereas the log-SPA implementation uses message additions. The latter is more efficient in fixed point implementations, as fixed point multiplications can take up many clock cycles compared to additions. Additions in the SPA are replaced by the Jacobi logarithm [9] which can be approximated by a very simple function, resulting in the max-log-SPA. Finally, the log-SPA implementation requires no normalization step [4]. A fortiori, SPA decoding over more general Galois fields suffers from the very same drawbacks. We note that a log-domain approach for non-binary turbo codes was investigated in [10].

In this contribution we derive the log-SPA decoding scheme for general non-binary LDPC codes based on LLRs. This algorithm is mathematically equivalent to the original decoding algorithm. Additionally, we investigate the performance degradation of a suboptimal variant (max-log-SPA) through computer simulations. Comparisons of the computational complexity and memory requirements of the various algorithms are carried out.

II. LDPC codes and belief propagation

A. Encoding

LDPC codes are linear block codes. They are defined by a very sparse parity check matrix in a finite Galois field $GF(q)$. This matrix $\mathbf{H}$ consists of M mutually independent rows and N columns. From $\mathbf{H}$, a systematic $N \times K$ (with $K = N - M$) generator matrix $\mathbf{G}^T$ can be constructed such that the rows of $\mathbf{H}$ generate the null-space of $\mathbf{G}^T$, i.e. $\mathbf{H}\mathbf{G}^T = \mathbf{0}$. In this contribution we consider irregular LDPC codes over binary extension fields: $GF\left(q = 2^b\right)$, although the proposed algorithm can be applied to more general fields. A block of Kb bits is converted to a sequence of K $GF\left(2^b\right)$ elements according to some mapping, $\varphi : \left(GF(2)\right)^b \rightarrow GF\left(2^b\right)$. The obtained information word, $\mathbf{b} \in \left(GF\left(2^b\right)\right)^K$, is encoded using the generator matrix, resulting in a codeword $\mathbf{c} \in \mathcal{C}$, with $\mathcal{C}$ denoting the set of codewords. Clearly, $\mathcal{C}$ is a linear subset of $\left(GF\left(2^b\right)\right)^N$. In matrix notation, this means:

$$\mathbf{c} = \mathbf{G}^T\mathbf{b}.$$

It can easily be shown that

$$\mathbf{c} \in \mathcal{C} \Leftrightarrow (\mathbf{s} \doteq) \, \mathbf{H}\mathbf{c} = \mathbf{0}. \tag{1}$$

The elements of $\mathbf{s}$ and $\mathbf{c}$ are referred to as the *checks* and *variables*, respectively. If a given check is zero, the participating variables satisfy that particular check. The codeword is now mapped to a signalling constellation, Ω, resulting in a vector $\mathbf{t}$**.** We will assume BPSK signalling, although extensions to higher order constellations are straightforward [11]. Thus, $\psi : GF(q) \rightarrow \Omega^b$, with $\Omega = \{-1, +1\}$ such that $\psi(c_k) = \left[t_{kb}, \ldots, t_{(k+1)b-1}\right]^T$ and $t_i \in \Omega$.

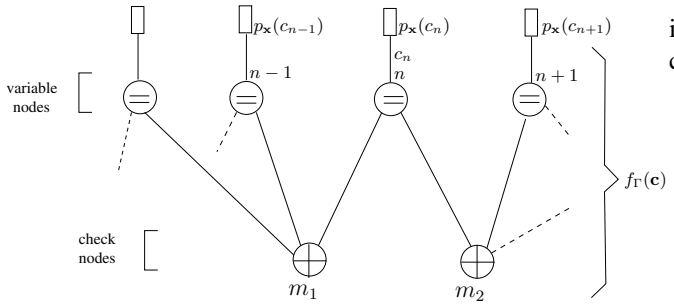

Fig. 1. part of a factor graph for an LDPC code over $GF(q)$

When the vector $\mathbf{t}$ is transmitted over an AWGN channel, it is corrupted by noise, yielding a received vector, $\mathbf{x}$:

$$\mathbf{x} = \mathbf{t} + \mathbf{n}$$

with $\mathbf{n}$ a real vector of independent white gaussian noise samples with power $\sigma^2 = N_0/(2E_bK/N)$ where E_b/N_0 is the SNR per information bit and K/N is the code rate.

B. Decoding

Although there exists a general class of decoding algorithms for LDPC codes, known as message-passing algorithms, we focus on the sum-product algorithm (SPA)[12]. This algorithm is based on a factor graph representation of the sparse parity check matrix [13]. Given some $\mathbf{H}$, the corresponding factor graph Γ consists of N variable nodes, associated with the N variables, and M check nodes, associated with the M checks. When $H_{m,n} \neq 0$, variable n is checked by check m. Hence, in Γ, check node m and variable node n are connected. This is shown in Fig. 1. Each node can be seen as a binary-valued function. For instance, the value of the function corresponding to check node m is 1 iff the adjacent variables satisfy that check, i.e., when the check $s_m = 0$. The function corresponding to variable node n assures consistency between different check nodes (hence the notation "=" in the variable nodes in Fig. 1). Consequently, the graph Γ can be seen binary-valued function $f_\Gamma(\mathbf{c})$: it evaluates to 1 iff all the checks are satisfied, i.e., $\mathbf{c} \in C$.

The joint probability density function (pdf) of $\mathbf{c}$ and $\mathbf{x}$ can be written as

$$\begin{aligned} p(\mathbf{c},\mathbf{x}) &\propto p(\mathbf{x}|\,\mathbf{c})\,p(\mathbf{c}) \\ &\propto p(\mathbf{x}|\,\mathbf{c})\,f_\Gamma(\mathbf{c}) \\ &= \prod_n p_{\mathbf{x}}(c_n)\,f_\Gamma(\mathbf{c}) \end{aligned}$$

with $p_{\mathbf{x}}(c_n) = p\left(x_{nb},\ldots,x_{b(n+1)+1}\middle|\, c_n\right)$. Hence, $p(\mathbf{c},\mathbf{x})$ can be represented by the factor graph in Fig. 1. We are interested in computing the a posteriori pdf of the variable $p(c_n|\,\mathbf{x})$. This can done efficiently using the sum-product algorithm by passing messages (pdfs) over the edges of the factor graph. The details of this algorithm are described in [13]. For LDPC codes, the message passing algorithms consists of the following steps:

1) Initialization step: variable nodes are initialized with the belief of the corresponding variable, based solely on the received vector $\mathbf{x}$.
2) Tentative decoding: variable node n computes, based on all the information it has available (i.e., from the channel vector $\mathbf{x}$ and messages from adjacent check nodes), the most likely value of variable n, $\hat{c}_n$. If the decoded word satisfies all checks ($\mathbf{H}\hat{\mathbf{c}} = \mathbf{0}$), the decoding algorithm is halted.
3) Horizontal step: a message (denoted by $\mathbf{L}(m \leftarrow n)$) is passed from variable node n to check node m, expressing the belief of the n-th variable, given all the information from *all* connected check nodes, *except* check node m itself.
4) Vertical step: each check node m sends a message (denoted by $\mathbf{L}(m \rightarrow n)$) to adjacent variable node n, reflecting the belief of the n-th variable, given all the information from the channel *and* all variable nodes connected to check node m, *except* variable node n itself. Go to step (2).

A decoding failure is declared if after a fixed number of iterations no valid codeword has been produced in step 2. The vague terms "belief" and "information" in the sum-product algorithm are expressed in terms of probabilities. When we use LLRs instead of probabilities, we end up with a log-domain version of the sum-product algorithm. In the next sections we will describe how this decoding algorithm may be expressed elegantly in the log-domain.

III. Log-likelihood algebra over binary extension fields

In this section we describe how probabilities of operations on random variables (RV) in $GF(q)$ can be converted to operations on the corresponding LLRs of the same RV. Computational complexity of these operations will be measured with the number of message operations (i.e., real-valued additions, multiplications, etc.). We assume that results of additions, multiplications, etc. of elements in $GF(q)$ are stored in a look-up table. This only requires a few small tables of size q^2. We denote the Galois field $GF(q)$ without the zero element by $GF_0(q) = \{\alpha_1,\ldots,\alpha_{q-1}\}$. We first introduce the notion of a LLR vector (LLRV) of v, a RV over $GF(q)$:

$$\mathbf{L}(v_1) = [L(v_1=\alpha_1)\ldots L(v_1=\alpha_{q-1})]^T$$

where

$$L(v=\alpha_i) = \ln\frac{P(v=\alpha_i)}{P(v=0)}.$$

with $P(v=\alpha_i)$ denoting the probability that v takes on the value α_i. Suppose we are given the LLRV of v_1 and v_2 (abbreviated by $\mathbf{L}_1$ and $\mathbf{L}_2$, respectively) and two elements in $GF_0(q)$: A_1 and A_2. In Appendix A we show that

$$\mathbf{L}(A_1v_1 + A_2v_2) = \boxplus(\mathbf{L}_1,\mathbf{L}_2,A_1,A_2) \qquad (2)$$

	+	*	max^*
SPA(H)	$(3u-4)\,Mq\,(q-1)$	$(3u-4)\,Mq^2$	0
SPA(V)	$uM\,(q-1)$	$Mtuq$	0
log-SPA(H)	$2\,(3u-4)\,M\,(q-1)^2$	0	$2\,(3u-4)\,M\,(q-1)^2$
log-SPA(V)	$uM\,(t-1)\,(q-1)$	0	0

TABLE I

NUMBER OF OPERATIONS ON MESSAGES PER ITERATION OF SPA AND LOG-SPA DECODING FOR (H)ORIZONTAL AND (V)ERTICAL STEP

where $\boxplus\,(\mathbf{L}_1, \mathbf{L}_2, A_1, A_2)$ is a simple extension of the box-plus operator introduced in [14] for $GF\,(2)$: $L\,(v_1+v_2) = L\,(v_1) \boxplus L\,(v_1)$. This latter expression, as well as (2), can be expressed in terms of the Jacobi logarithm:

$$max^*\,(x_1, x_2) \;\doteq\; \ln\left(e^{x_1}+e^{x_2}\right). \tag{3}$$

It can easily be seen that $max^*\,(x_1, x_2, x_3) = max^*\,(max^*\,(x_1, x_2), x_3)$, so $\boxplus\,(\mathbf{L}_1, \mathbf{L}_2, A_1, A_2)$ can be computed recursively. The computation of $\mathbf{L}\,(A_1 v_1 + A_2 v_2)$ requires $2\,(q-1)^2$ message additions and $2\,(q-1)^2$ max^* operations. Note that max^* can be expressed as

$$max^*\,(x_1, x_2) = \max\,(x_1, x_2) + \ln\left(1+e^{-|x_1-x_2|}\right).$$

Consequently, max^* requires a maximization, corrected by a term $(\ln\,(1+\cdot))$. The correction term may be stored, without any performance loss, in a small look-up table as a function of $|x_1 - x_2|$ [9]. Hence, an efficient implementation of max^* can be realized by one comparison, two message additions and one table look-up. Note that a probability-based computation of the distribution of $(A_1 v_1 + A_2 v_2)$ requires $q\,(q-1)$ additions and q^2 multiplications on messages.

IV. LOG-SPA DECODING OF LDPC CODES

With each check node (m, $1 \le m \le M$) and variable node (n, $1 \le n \le N$) we associate two kinds of messages:

- LLRV messages from check node m to variable node n: $\mathbf{L}\,(m \to n)$, representing the LLRV of the n-th variable, given the LLRV of all variables checked by the m-th check, except the n-th variable itself.
- LLRV messages from variable node n to check node m: $\mathbf{L}\,(m \leftarrow n)$, representing the LLRV of the n-th variable, given the LLRV of all checks involving variable n, except check m itself.

We further denote by $\mathcal{M}\,(n)$, the set of check nodes connected to variable node n. Similarly, we denote by $\mathcal{N}\,(m)$ the set of variable nodes connected to check node m. For each check node m, we order $\mathcal{N}\,(m)$: $n_{m,0} < n_{m,1} < \ldots < n_{m,end-1}$.

(1) Demapping and initialization : First, we determine the LLRV corresponding to variable node n, according to the channel model. We refer again to the Appendix (section B) for more details. It turns out that for BPSK signalling the i-th component is given by

$$\mathbf{L}_{ch}\,(c_n)_i \;=\; \sum_{j:\psi^{-1}(\alpha_i)_j=+1} \frac{2x_{nb+j}}{\sigma^2}. \tag{4}$$

We now initialize, for $1 \le n \le N$ and $1 \le m \le M$ such that $H_{m,n} \neq 0$, the LLRV messages as follows

$$\begin{aligned} \mathbf{L}\,(m \leftarrow n) &= \mathbf{L}_{ch}\,(c_n) \\ \mathbf{L}\,(m \to n) &= \mathbf{0}. \end{aligned}$$

(2) Tentative decoding: We now compute the a posteriori LLRV for each variable, denoted by $\mathbf{L}_{post}\,(c_n)$, $1 \le n \le N$:

$$\mathbf{L}_{post}\,(c_n) = \mathbf{L}_{ch}\,(c_n) + \sum_{j \in \mathcal{M}(n)} \mathbf{L}\,(j \to n). \tag{5}$$

From this LLRV we can easily determine the most likely value of the n-th variable. We then verify the M checks according to (1). If all checks are satisfied, we stop decoding.

(3) Horizontal step: The message from variable node n to check node m is simply given by (for $1 \le n \le N$ and $m \in \mathcal{M}\,(n)$):

$$\mathbf{L}\,(m \leftarrow n) = \mathbf{L}_{ch}\,(c_n) + \sum_{j \in \mathcal{M}(n)\setminus m} \mathbf{L}\,(j \to n). \tag{6}$$

(4) Vertical step: For each check node m and adjacent variable node $n_{m,k}$, we introduce new RVs in $GF\,(q)$: $\sigma_{m,n_{m,l}} = \sum_{j \le l} H_{m,n_{m,j}} c_{n_{m,j}}$ and $\rho_{m,n_{m,l}} = \sum_{j \ge l} H_{m,n_{m,j}} c_{n_{m,j}}$. Following a line of reasoning similar to [4], it can easily be shown that the distributions of $\sigma_{m,n_{m,l}}$ and $\rho_{m,n_{m,l}}$ can be computed recursively according to:

$$\begin{cases} \mathbf{L}\left(\sigma_{m,n_{m,l}}\right) = \mathbf{L}\left(\sigma_{m,n_{m,l-1}} + H_{m,n_{m,l}} c_{n_{m,l}}\right) \\ \mathbf{L}\left(\rho_{m,n_{m,l}}\right) = \mathbf{L}\left(\rho_{m,n_{m,l+1}} + H_{m,n_{m,l}} c_{n_{m,l}}\right) \end{cases}$$

where the LLRV of $c_{n_{m,l}}$ is given by $\mathbf{L}\,(m \leftarrow n_{m,l})$. The message from check node m to variable node $n_{m,k}$ is then given by (for $1 \le m \le M$, $n \in \mathcal{N}\,(m)$):

$$\begin{aligned} &\mathbf{L}\,(m \to n_{m,k}) = \\ &\mathbf{L}\left(H^{-1}_{m,n_{m,k}} \sigma_{m,n_{m,k-1}} + H^{-1}_{m,n_{m,k}} \rho_{m,n_{m,k+1}}\right). \end{aligned} \tag{7}$$

These LLRV can all be computed efficiently by means of the box-plus operator (2). The messages (7) are used to update the posteriori LLRV (5). The vertical step is the most time-consuming step in the decoding algorithm.

V. Computational complexity and BER performance

In this section we compare three decoding algorithms for LDPC codes over $GF(q)$: the SPA algorithm from [4], the log-SPA algorithm as described above, and the reduced complexity max-log-SPA algorithm. In the latter, $max^*()$ is approximated by $\max()$.

Fig. 2 shows the BER performance for a rate $1/2$ LDPC code from [15] over $GF(8)$ with $N = 204$. The decoding process is halted after a maximum of 100 iterations. As expected, the log-SPA algorithm yields the same BER performance as the conventional SPA algorithm. The max-log-SPA version suffers a performance penalty of about 0.5 dB, which is similar to turbo-coded systems [16]. It should be noted that for all algorithms and all SNRs, all errors were detected errors (i.e., the maximum number of decoding iterations was reached and no codeword was found).

When comparing complexity, we first observe that the memory requirement for all three decoding algorithms is roughly the same: decoding in the log-domain results in a reduction of a factor $q/(q-1)$ in storage. In Table I we compare the computational complexity of the decoding algorithms for the horizontal (H) and the vertical step (V). We have determined the number of message additions, multiplications and max^* operations. The total number of operations is determined by the field size (q), the number of variable nodes (N), the number of check nodes (M), the mean column weight of $\mathbf{H}$ ($t \geq 2$) and the mean row weight of $\mathbf{H}$ ($u \geq 2$). Note that each $max^*()$ operation corresponds to one $\max()$ operation, two additions and one table look-up. When using the reduced complexity max-log-SPA algorithm, $max^*()$ may simply be replaced by $\max()$ in Table I to determine the number of computations. The SPA and log-SPA algorithm both require the same order of message additions ($\mathcal{O}\left(q^2Mu\right)$ per iteration). When implemented on a fixed point architecture, the former algorithm also needs $\mathcal{O}\left(q^2Mu\right)$ computationally intensive fixed point multiplications per iteration. The latter algorithm avoids multiplications but uses $\mathcal{O}\left(q^2Mu\right)$ much less demanding max^* operations per iteration. Therefore, the log-SPA algorithm has the smaller computational complexity in a fixed point architecture. We note that in [5] a decoding algorithm for high-rate codes using fast fourier transforms (FFT) was proposed of $\mathcal{O}\left(Mqu\left(\log_2 q + u\right)\right)$. However, this FFT-based algorithm did not operate on LLRs, and it is not suitable for low-rate codes.

VI. Conclusions and remarks

In this contribution we have introduced a log-domain version of the sum-product decoding algorithm (SPA) for LDPC codes over $GF(q)$. A log-domain implementation has several advantages as far as practical implementation is concerned. We have compared the BER performance and computational complexity of the SPA, the log-domain version (log-SPA) and a sub-optimal implementation (max-log-SPA). As log-domain SPA and SPA are mathematically equivalent, they have identical BER performance. This has been confirmed by computer simulations. Max-log-SPA gives rise to a small BER degradation (about 0.5 dB). As the log-SPA requires no message multiplications, it may constitute a considerable saving in computational complexity as compared to the SPA.

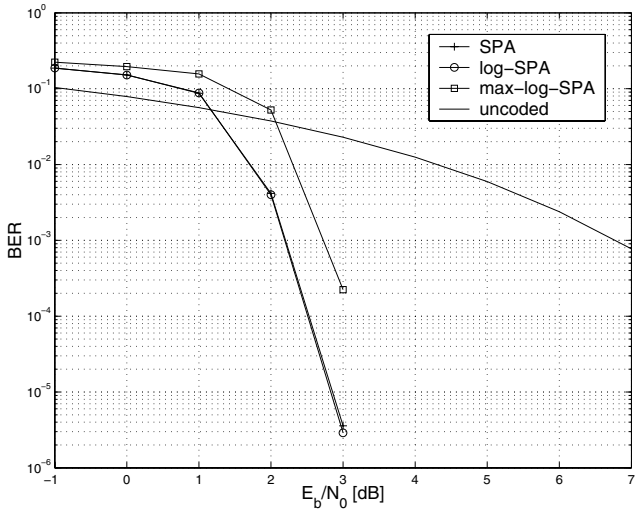

Fig. 2. BER performance for an LDPC code over $GF(8)$

Acknowledgement

This work has been supported by the Interuniversity Attraction Poles Program P5/11 - Belgian Science Policy.

Appendix

A. Log-likelihood algebra

Suppose v_1 and v_2 are RVs in $GF(q)$; A_1 and A_2 are elements in $GF_0(q)$. Given the LLRV of v_1 and v_2, abbreviated by $\mathbf{L}_1$ and $\mathbf{L}_2$ respectively, we now determine $\mathbf{L}(A_1v_1 + A_2v_2)$. Its i-th component is given by, with $L_k(x) \doteq L(v_k = x)$, $k = 1, 2$:

$$
\begin{aligned}
&\mathbf{L}(A_1v_1 + A_2v_2)_i \\
&= \ln \frac{P(A_1v_1 + A_2v_2 = \alpha_i)}{P(A_1v_1 + A_2v_2 = 0)} \\
&= \ln \frac{\sum_{x \in GF(q)} P(v_1 = x)\, P\left(v_2 = A_2^{-1}(\alpha_i + xA_1)\right)}{\sum_{x \in GF(q)} P(v_1 = x)\, P\left(v_2 = A_2^{-1}A_1x\right)} \\
&= \ln \frac{\sum_{x \in GF(q)} \frac{P(v_1=x)P\left(v_2=A_2^{-1}(\alpha_i+xA_2)\right)}{P(v_1=0)P(v_2=0)}}{1 + \sum_{x \in GF_0(q)} \frac{P(v_1=x)P\left(v_2=A_2^{-1}A_1x\right)}{P(v_1=0)P(v_2=0)}}.
\end{aligned}
$$

This can be re-written using $\mathbf{L}_1$ and $\mathbf{L}_2$:

$$
\begin{aligned}
&\mathbf{L}\left(A_1 v_1 + A_2 v_2\right)_i \\
&= \ln\left(e^{L_1\left(A_1^{-1}\alpha_i\right)} + e^{L_2\left(A_2^{-1}\alpha_i\right)} \right. \\
&\left. + \sum_{x \in GF_0(q)\setminus\left\{\alpha_i A_1^{-1}\right\}} e^{L_1(x)+L_2\left(A_2^{-1}(\alpha_i + xA_1)\right)} \right) \\
&- \ln\left(1 + \sum_{x \in GF_0(q)} e^{L_1(x)+L_2\left(A_2^{-1}A_1 x\right)} \right) \\
&\doteq \boxplus\left(\mathbf{L}_1, \mathbf{L}_2, A_1, A_2\right)_i .
\end{aligned}
$$

B. Soft demapping

We determine the i-th component of the LLRV corresponding to the n-th variable for an AWGN channel and BPSK signalling. These messages are the LLR equivalent of the probabilities $p_{\mathbf{x}}(c_n) = p\left(x_{nb}, \ldots, x_{b(n+1)+1} \middle| \mathbf{c}\right)$ in the factor graph from Fig. 1:

$$
\begin{aligned}
&\mathbf{L}_{ch}\left(c_n\right)_i \\
&= \ln \frac{P\left(\mathbf{x} \middle| c_n = \alpha_i\right)}{P\left(\mathbf{x} \middle| c_n = 0\right)} \\
&= \ln \frac{P\left(\mathbf{x} \middle| \psi\left(t_{nb}, t_{nb+1}, \ldots, t_{(n+1)b-1}\right) = \alpha_i\right)}{P\left(\mathbf{x} \middle| \psi\left(t_{nb}, t_{nb+1}, \ldots, t_{(n+1)b-1}\right) = 0\right)} \\
&\langle \text{AWGN channel} \rangle \\
&= \ln \frac{\prod_j P\left(x_{nb+j} \middle| t_{nb+j} = \left(\psi^{-1}\left(\alpha_i\right)\right)_j\right)}{\prod_j P\left(x_{nb+j} \middle| t_{nb+j} = \left(\psi^{-1}\left(0\right)\right)_j\right)} \\
&= \ln \prod_{j:\left(\psi^{-1}(\alpha_i)\right)_j = +1} \frac{P\left(x_{nb+j} \middle| t_{nb+j} = +1\right)}{P\left(x_{nb+j} \middle| t_{nb+j} = -1\right)} \\
&= \sum_{j:\left(\psi^{-1}(\alpha_i)\right)_j = +1} \frac{2x_{nb+j}}{\sigma^2}.
\end{aligned}
$$

REFERENCES

[1] R.G. Gallager. "Low density parity-check codes". *IRE Trans. Inform. Theory*, IT-8:pp. 21–29, Jan. 1962.

[2] D.J.C. MacKay. "Good error-correcting codes based on very sparse matrices". *IEEE Trans. Inform. Theory*, 45(2):pp. 399–431, March 1999.

[3] T. Richardson, M. Shokrollahi and R. Urbanke. "Design of capacity approaching low-density parity-check codes". *IEEE Trans. Inform. Theory*, 47(2):619–637, February 2001.

[4] M.C. Davey and D.J.C. MacKay. "Low density parity check codes over GF(q)". *IEEE Comm. Letters*, 2(6):pp.165–167, June 1998.

[5] H. Song and J.R. Cruz. "Reduced complexity decoding of Q-ary LDPC codes for magnetic recording". *IEEE Trans. on Magnetics*, 39(2):1081–1087, March 2003.

[6] D. J. C. MacKay, S. T. Wilson, and M. C. Davey. "Comparison of Constructions of Irregular Gallager Codes". *IEEE Transactions on Communications*, 47(10):1449–1454, October 1999.

[7] L. Ping and W.K. Leung. "Decoding low density parity check codes with finite quantization bits". *IEEE Comm. Letters*, 4(2):pp.62–64, February 2000.

[8] H. Wymeersch, H. Steendam and M. Moeneclaey. "Computational complexity and quantization effects of decoding algorithms of LDPC codes over $GF(q)$". In *Proc. ICASSP*, Montreal, Canada, May 2004.

[9] X. Hu, E. Eleftheriou, D.-M. Arnold and A. Dholakia. "Efficient implementations of the sum-product algorithm for decoding LDPC codes". In *Proc. IEEE Globecom*, 2001.

[10] J. Berkmann. "On turbo decoding of nonbinary codes". *IEEE Comm. Letters*, 2(4):94–96, April 1998.

[11] S. ten Brink, J. Speidel and J. C. Yan. "Iterative demapping and decoding for multilevel modulation". In *IEEE GLOBECOM'98*, 1998.

[12] J. Pearl. *"Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference"*. Morgan Kaufmann, San Mateo, 1988.

[13] F. Kschinschang, B. Frey and H.-A. Loeliger. "Factor graphs and the sum-product algorithm". *IEEE Trans. Inform. Theory*, 47(2):pp.498–519, February 2001.

[14] J. Hagenauer, E. Offer, C. Méasson and M. Mörz. "Decoding and equalization with analog non-linear networks". *European Trans. Comm.*, 10:pp.659–680, November 1999.

[15] D.J.C. MacKay. "Online database of low-density parity check codes". http://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

[16] M.C. Valenti. "An efficient software radio implementation of the UMTS turbo codec". In *Proc. IEEE PIMRC*, pages 108–113, San Diego (USA), September 2001.