Low-Density Lattice Codes

Naftali Sommer, *Senior Member, IEEE*, Meir Feder, *Fellow, IEEE*, and Ofir Shalvi, *Member, IEEE*

***Abstract*—Low-density lattice codes (LDLC) are novel lattice codes that can be decoded efficiently and approach the capacity of the additive white Gaussian noise (AWGN) channel. In LDLC a codeword $\underline{x}$ is generated directly at the n-dimensional Euclidean space as a linear transformation of a corresponding integer message vector $\underline{b}$, i.e., $\underline{x} = \boldsymbol{G}\underline{b}$, where $\boldsymbol{H} = \boldsymbol{G}^{-1}$ is restricted to be sparse. The fact that $\boldsymbol{H}$ is sparse is utilized to develop a linear-time iterative decoding scheme which attains, as demonstrated by simulations, good error performance within $\sim$0.5 dB from capacity at block length of n =100,000 symbols. The paper also discusses convergence results and implementation considerations.**

***Index Terms*—Iterative decoding, lattice codes, lattices, low-density parity-check (LDPC) code.**

I. Introduction

If we take a look at the evolution of codes for binary or finite alphabet channels, it was first shown [1] that channel capacity can be achieved with long random codewords. Then, it was found out [2] that capacity can be achieved via a simpler structure of linear codes. Then, specific families of linear codes were found that are practical and have good minimum Hamming distance (e.g., convolutional codes, cyclic block codes, specific cyclic codes such as Bose–Chaudhuri–Hocquenghen (BCH) and Reed–Solomon codes [4]). Later, capacity achieving schemes were found, which have special structures that allow efficient iterative decoding, such as low-density parity-check (LDPC) codes [5] or turbo codes [6].

If we now take a similar look at continuous alphabet codes for the additive white Gaussian noise (AWGN) channel, it was first shown [3] that codes with long random Gaussian codewords can achieve capacity. Later, it was shown that lattice codes can also achieve capacity ([7]–[12]). Lattice codes are clearly the Euclidean space analog of linear codes. Similar to binary codes, we could expect that specific practical lattice codes will then be developed. However, there was almost no further progress from that point. Specific lattice codes that were found were based on fixed dimensional classical lattices [20] or based on algebraic error correcting codes [13], [14], and even on LDPC codes [15], but no significant effort was made in designing lattice codes directly in the Euclidean space or in finding specific capacity achieving lattice codes. Practical coding schemes for the AWGN channel were based on finite alphabet codes.

In [16], "signal codes" were introduced. These are lattice codes, designed directly in the Euclidean space, where the information sequence of integers $i_n, n = 1, 2, \ldots$ is encoded by convolving it with a fixed signal pattern $g_n, n = 1, 2, \ldots, d$. Signal codes are clearly analogous to convolutional codes, and in particular can work at the AWGN channel cutoff rate with simple sequential decoders. In [17] it is also demonstrated that signal codes can work near the AWGN channel capacity with more elaborated bi-directional decoders. Thus, signal codes provided the first step toward finding effective lattice codes with practical decoders.

Inspired by LDPC codes and in the quest of finding practical capacity achieving lattice codes, we propose in this work low-density lattice codes (LDLC). We show that these codes can approach the AWGN channel capacity with iterative decoders whose complexity is linear in block length. In recent years several schemes were proposed for using LDPC over continuous valued channels by either multilevel coding [19] or by nonbinary alphabet (e.g., [18]). Unlike these LDPC-based schemes, in LDLC both the encoder and the channel use the same real algebra which is natural for the continuous-valued AWGN channel. This feature also simplifies the convergence analysis of the iterative decoder.

The outline of this paper is as follows. LDLC are first defined in Section II. The iterative decoder is then presented in Section III, followed by convergence analysis of the decoder in Section IV. Then, Section V describes how to choose the LDLC code parameters, and Section VI discusses implementation considerations. The computational complexity of the decoder is then discussed in Section VII, followed by a brief description of encoding and shaping in Section VIII. Simulation results are finally presented in Section IX.

II. Basic Definitions and Properties

A. Lattices and Lattice Codes

An n dimensional lattice in $\mathbb{R}^m$ is defined as the set of all linear combinations of a given basis of n linearly independent vectors in $\mathbb{R}^m$ with integer coefficients. The matrix $\boldsymbol{G}$, whose columns are the basis vectors, is called a generator matrix of the lattice. Every lattice point is therefore of the form $\underline{x} = \boldsymbol{G}\underline{b}$, where $\underline{b}$ is an n-dimensional vector of integers. The Voronoi cell of a lattice point is defined as the set of all points that are closer to this point than to any other lattice point. The Voronoi cells of all lattice points are congruent, and for square $\boldsymbol{G}$ the volume of the Voronoi cell is equal to $\det(\boldsymbol{G})$. In the sequel $\boldsymbol{G}$ will be used to denote both the lattice and its generator matrix.

Manuscript received October 29, 2006 ; revised August 30, 2007. The material in this paper was presented in part at the IEEE International Symposium on Information Theory (ISIT), Seattle, WA, July 2006 and in part at the Inauguration of the UCSD Information Theory and Applications Center, San Diego, CA, February 2006.

N. Sommer and O. Shalvi are with the Department of Electrical Engineering-Systems, Tel-Aviv University, Tel-Aviv, Israel. They are also with Anobit Technologies, Herzlia, Israel.

M. Feder is with the Department of Electrical Engineering-Systems, Tel-Aviv University, Tel-Aviv, Israel.

Color versions of Figures 3 and 4 in this paper are available online at http://ieeexplore.ieee.org.

Communicated by T. J. Richardson, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2008.917684

A lattice code of dimension n is defined by a (possibly shifted) lattice $\boldsymbol{G}$ in $\mathbb{R}^m$ and a shaping region $B \subset \mathbb{R}^m$, where the codewords are all the lattice points that lie within the shaping region B. Denote the number of these codewords by N. The average transmitted power (per channel use, or per symbol) is the average energy of all codewords, divided by the codeword length m. The information rate (in bits/symbol) is $\log_2(N)/m$. Throughout this paper we shall assume that $\boldsymbol{G}$ is square (i.e., $n = m$), but the results are easily extended to the nonsquare case.

When using a lattice code for the AWGN channel with power limit P and noise variance σ^2, the maximal information rate is limited by the capacity $\frac{1}{2}\log_2(1+\frac{P}{\sigma^2})$. Poltyrev [21] considered the AWGN channel without restrictions. If there is no power restriction, code rate is a meaningless measure, since it can be increased without limit. Instead, it was suggested in [21] to use the measure of constellation density, leading to a generalized definition of the capacity as the maximal possible codeword density that can be recovered reliably. When applied to lattices, the generalized capacity implies that there exists a lattice $\boldsymbol{G}$ of high enough dimension n that enables transmission with arbitrary small error probability, if and only if $\sigma^2 < \frac{\sqrt[n]{|\det(\boldsymbol{G})|^2}}{2\pi e}$. A lattice that achieves the generalized capacity of the AWGN channel without restrictions, also achieves the channel capacity of the power constrained AWGN channel, with a properly chosen spherical shaping region (see also [12]).

In the rest of this work we shall concentrate on the lattice design and the lattice decoding algorithm, and not on the shaping region or shaping algorithms. We shall use lattices with $\det(\boldsymbol{G}) = 1$, where analysis and simulations will be carried for the AWGN channel without restrictions. A capacity achieving lattice will have small error probability for noise variance σ^2 which is close to the theoretical limit $\frac{1}{2\pi e}$.

B. Syndrome and Parity Check Matrix for Lattice Codes

A binary (n, k) error correcting code is defined by its $n \times k$ binary generator matrix $\boldsymbol{G}$. A binary information vector $\underline{b}$ with dimension k is encoded by $\underline{x} = \boldsymbol{G}\underline{b}$, where calculations are performed in the finite field GF(2). The parity check matrix $\boldsymbol{H}$ is an $(n-k) \times n$ matrix such that $\underline{x}$ is a codeword if and only if $\boldsymbol{H}\underline{x} = \underline{0}$. The input to the decoder is the noisy codeword $\underline{y} = \underline{x} + \underline{e}$, where $\underline{e}$ is the error sequence and addition is done in the finite field. The decoder typically starts by calculating the syndrome $\underline{s} = \boldsymbol{H}\underline{y} = \boldsymbol{H}(\underline{x} + \underline{e}) = \boldsymbol{H}\underline{e}$ which depends only on the noise sequence and not on the transmitted codeword.

We would now like to extend the definitions of the parity check matrix and the syndrome to lattice codes. An n-dimensional lattice code is defined by its $n \times n$ lattice generator matrix $\boldsymbol{G}$. Every codeword is of the form $\underline{x} = \boldsymbol{G}\underline{b}$, where $\underline{b}$ is a vector of integers. Therefore, $\boldsymbol{G}^{-1}\underline{x}$ is a vector of integers for every codeword $\underline{x}$. We define the parity check matrix for the lattice code as $\boldsymbol{H} \triangleq \boldsymbol{G}^{-1}$. Given a noisy codeword $\underline{y} = \underline{x} + \underline{w}$ (where w is the additive noise vector, e.g., AWGN, added by real arithmetic), we can then define the syndrome as $\underline{s} \triangleq frac\{\boldsymbol{H}\underline{y}\}$, where $frac\{x\}$ is the fractional part of x, defined as $frac\{x\} = x - \lfloor x \rceil$, where $\lfloor x \rceil$ denotes the nearest integer to x. The syndrome $\underline{s}$ will be zero if and only if $\underline{y}$ is a lattice point, since $\boldsymbol{H}\underline{y}$ will then be a vector of integers with zero fractional part. For a noisy codeword, the syndrome will equal $\underline{s} = frac\{\boldsymbol{H}\underline{y}\} = frac\{\boldsymbol{H}(\underline{x}+\underline{w})\} = frac\{\boldsymbol{H}\underline{w}\}$ and therefore will depend only on the noise sequence and not on the transmitted codeword, as desired.

Note that the above definitions of the syndrome and parity check matrix for lattice codes are consistent with the definitions of the dual lattice and the dual code[20]: the dual lattice of a lattice $\boldsymbol{G}$ is defined as the lattice with generator matrix $\boldsymbol{H} = \boldsymbol{G}^{-1}$, where for binary codes, the dual code of $\boldsymbol{G}$ is defined as the code whose generator matrix is $\boldsymbol{H}$, the parity check matrix of $\boldsymbol{G}$.

C. Low-Density Lattice Codes (LDLC)

We shall now turn to the definition of the codes proposed in this paper—LDLC.

Definition 1 (LDLC): An n dimensional LDLC is an n-dimensional lattice code with a nonsingular lattice generator matrix $\boldsymbol{G}$ satisfying $|\det(\boldsymbol{G})| = 1$, for which the parity check matrix $\boldsymbol{H} = \boldsymbol{G}^{-1}$ is sparse. The ith row degree r_i, $i = 1, 2, \ldots, n$ is defined as the number of nonzero elements in row i of $\boldsymbol{H}$, and the ith column degree c_i, $i = 1, 2, \ldots, n$ is defined as the number of nonzero elements in column i of $\boldsymbol{H}$.

Note that in binary LDPC codes, the code is completely defined by the locations of the nonzero elements of $\boldsymbol{H}$. In LDLC there is another degree of freedom since we also have to choose the *values* of the nonzero elements of $\boldsymbol{H}$.

Definition 2 (Regular LDLC): An n dimensional LDLC is regular if all the row degrees and column degrees of the parity check matrix are equal to a common degree d.

Definition 3 (Latin Square LDLC): An n dimensional regular LDLC with degree d is called "Latin square LDLC" if every row and column of the parity check matrix $\boldsymbol{H}$ has the same d nonzero values, except for a possible change of order and random signs. The sorted sequence of these d values $h_1 \geq h_2 \geq \cdots \geq h_d > 0$ will be referred to as the generating sequence of the Latin square LDLC.

For example, the matrix

$$\boldsymbol{H} = \begin{pmatrix} 0 & -0.8 & 0 & -0.5 & 1 & 0 \\ 0.8 & 0 & 0 & 1 & 0 & -0.5 \\ 0 & 0.5 & 1 & 0 & 0.8 & 0 \\ 0 & 0 & -0.5 & -0.8 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0.5 & 0.8 \\ 0.5 & -1 & -0.8 & 0 & 0 & 0 \end{pmatrix}$$

is a parity check matrix of a Latin square LDLC with lattice dimension $n = 6$, degree $d = 3$ and generating sequence $1, 0.8, 0.5$. This $\boldsymbol{H}$ should be further normalized by the constant $\sqrt[n]{|\det(\boldsymbol{H})|}$ in order to have $|\det(\boldsymbol{H})| = |\det(\boldsymbol{G})| = 1$, as required by Definition 1.

The bipartite graph of an LDLC is defined similarly to LDPC codes: it is a graph with variable nodes at one side and check nodes at the other side. Each variable node corresponds to a single element of the codeword $\underline{x} = \boldsymbol{G}\underline{b}$. Each check node corresponds to a check equation (a row of $\boldsymbol{H}$). A check equation

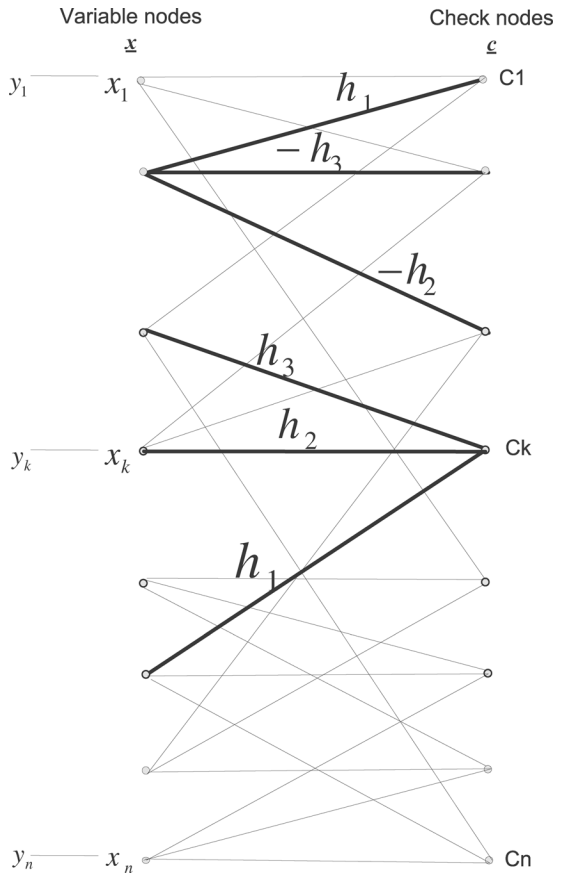

Fig. 1. The bipartite graph of an LDLC.

is of the form $\sum_k h_k x_{i_k} = integer$, where i_k denotes the locations of the nonzero elements at the appropriate row of $\boldsymbol{H}$, h_k are the values of $\boldsymbol{H}$ at these locations and the integer at the right hand side is unknown. An edge connects check node i and variable node j if and only if $H_{i,j} \neq 0$. This edge is assigned the value $H_{i,j}$. Fig. 1 illustrates the bipartite graph of a Latin square LDLC with degree 3. In the figure, every variable node x_k is also associated with its noisy channel observation y_k.

Finally, a k-loop is defined as a loop in the bipartite graph that consists of k edges. A bipartite graph, in general, can only contain loops with even length. Also, a two-loop, which consists of two parallel edges that originate from the same variable node to the same check node, is not possible by the definition of the graph. However, longer loops are certainly possible. For example, a four-loop exists when two variable nodes are both connected to the same pair of check nodes.

III. Iterative Decoding for the AWGN Channel

Assume that the codeword $\underline{x} = \boldsymbol{G}\underline{b}$ was transmitted, where $\underline{b}$ is a vector of integers. We observe the noisy codeword $\underline{y} = \underline{x} + \underline{w}$, where $\underline{w}$ is a vector of i.i.d Gaussian noise samples with common variance σ^2, and we need to estimate the integer valued vector $\underline{b}$. The maximum likelihood (ML) estimator is then $\hat{\underline{b}} = \arg\min_{\underline{b}} \|\underline{y} - \boldsymbol{G}\underline{b}\|^2$.

Our decoder will not estimate directly the integer vector $\underline{b}$. Instead, it will estimate the probability density function (pdf) of the codeword vector $\underline{x}$. Furthermore, instead of estimating the n-dimensional pdf of the whole vector $\underline{x}$, we shall estimate the n one-dimensional marginal pdfs for each of the components x_k of this vector (conditioned on the whole observation vector $\underline{y}$). In Appendix I it is shown that $f_{x_k|\underline{y}}(x_k|\underline{y})$ is a weighted sum of Dirac delta functions:

$$f_{x_k|\underline{y}}(x_k|\underline{y}) = C \cdot \sum_{\underline{l} \in \boldsymbol{G} \cap B} \delta(x_k - l_k) \cdot e^{-d^2(\underline{l},\underline{y})/2\sigma^2} \qquad (1)$$

where $\underline{l}$ is a lattice point (vector), l_k is its kth component, C is a constant independent of x_k and $d(\underline{l},\underline{y})$ is the Euclidean distance between $\underline{l}$ and $\underline{y}$. Direct evaluation of (1) is not practical, so our decoder will try to estimate $f_{x_k|\underline{y}}(x_k|\underline{y})$ (or at least approximate it) in an iterative manner.

Our decoder will decode to the infinite lattice, thus ignoring the shaping region boundaries. This approximate decoding method is no longer exact maximum likelihood decoding, and is usually denoted "lattice decoding" [12].

The calculation of $f_{x_k|\underline{y}}(x_k|\underline{y})$ is involved since the components x_k are not independent random variables (RVs), because $\underline{x}$ is restricted to be a lattice point. Following [5] we use a "trick" – we assume that the x_k's are independent, but add a condition that assures that $\underline{x}$ is a lattice point. Specifically, define $\underline{s} \triangleq \boldsymbol{H} \cdot \underline{x}$. Restricting $\underline{x}$ to be a lattice point is equivalent to restricting $\underline{s} \in \mathbb{Z}^n$. Therefore, instead of calculating $f_{x_k|\underline{y}}(x_k|\underline{y})$ under the assumption that $\underline{x}$ is a lattice point, we can calculate $f_{x_k|\underline{y}}(x_k|\underline{y}, \underline{s} \in \mathbb{Z}^n)$ and assume that the x_k are independent and identically distributed (i.i.d.) with a continuous pdf (that does not include Dirac delta functions). It still remains to set $f_{x_k}(x_k)$, the pdf of x_k. Under the i.i.d. assumption, the pdf of the codeword $\underline{x}$ is $f_{\underline{x}}(\underline{x}) = \prod_{k=1}^{n} f_{x_k}(x_k)$. As shown in Appendix II, the value of $f_{\underline{x}}(\underline{x})$ is not important at values of $\underline{x}$ which are not lattice points, but at a lattice point it should be proportional to the probability of using this lattice point. Since we assume that all lattice points are used equally likely, $f_{\underline{x}}(\underline{x})$ must have the same value at all lattice points. A reasonable choice for $f_{x_k}(x_k)$ is then to use a uniform distribution such that $\underline{x}$ will be uniformly distributed in an n-dimensional cube. For an exact ML decoder (that takes into account the boundaries of the shaping region), it is enough to choose the range of $f_{x_k}(x_k)$ such that this cube will contain the shaping region. For our decoder, that performs lattice decoding, we should set the range of $f_{x_k}(x_k)$ large enough such that the resulting cube will include all the lattice points which are likely to be decoded. The derivation of the iterative decoder shows that this range can be set as large as needed without affecting the complexity of the decoder.

The derivation in [5] further imposed the tree assumption. In order to understand the tree assumption, it is useful to define the tier diagram (also referred to as "the computation tree" [22]), which is shown in Fig. 2 for a regular LDLC with degree 3. Each vertical line corresponds to a check equation. The tier 1 nodes of x_1 are all the elements x_k that take place in a check equation with x_1. The tier 2 nodes of x_1 are all the elements that take place in check equations with the tier 1 elements of x_1, and so on. The tree assumption assumes that all the tree elements are distinct (i.e., no element appears in different tiers or twice in the same tier). This assumption simplifies the derivation, but in general, does not hold in practice, so our iterative

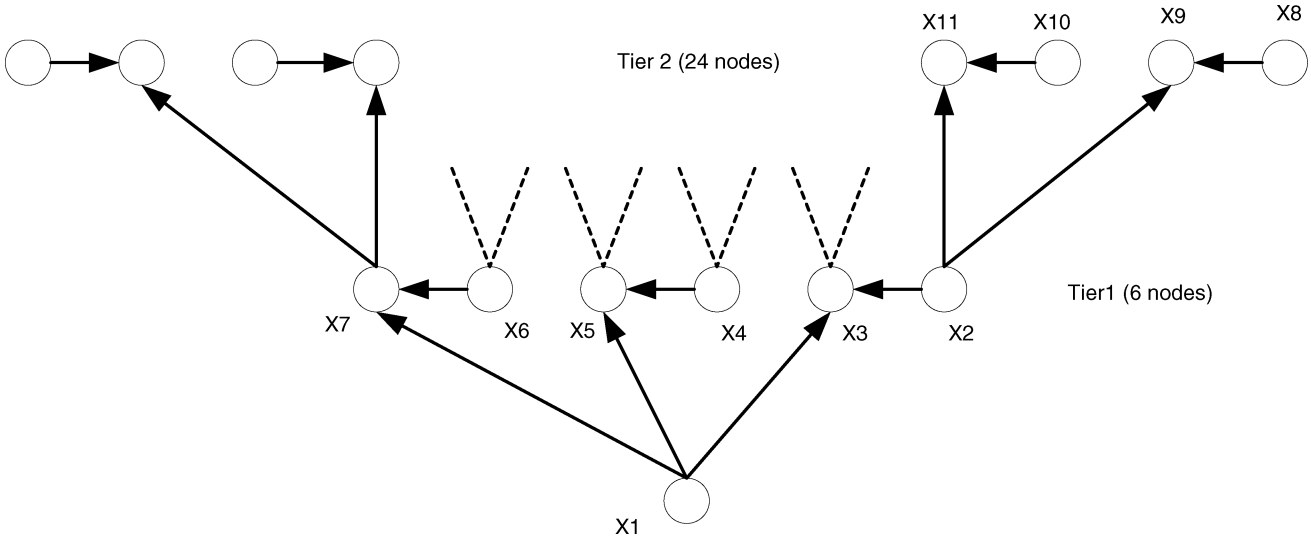

Fig. 2. Tier diagram.

algorithm is not guaranteed to converge to the exact solution (1) (see Section IV).

The detailed derivation of the iterative decoder (using the above "trick" and the tree assumption) is given in Appendix III. In Section III-A below, we present the final resulting algorithm. This iterative algorithm can also be developed using the principles of the sum-product algorithm, as explained after the algorithm specification.

A. The Iterative Decoding Algorithm

The iterative algorithm is most conveniently represented by using a message passing scheme over the bipartite graph of the code, similar to LDPC codes. The basic difference is that in LDPC codes the messages are scalar values (e.g., the log likelihood ratio of a bit), where for LDLC the messages are real functions over the interval $(-\infty, \infty)$. As in the decoding of LDPC codes, in each iteration the check nodes send messages to the variable nodes along the edges of the bipartite graph and vice versa. The messages sent by the check nodes are periodic extensions of pdfs. The messages sent by the variable nodes are pdfs.

LDLC iterative decoding algorithm:
Denote the variable nodes by $x_1, x_2, \ldots, x_n$ and the check nodes by $c_1, c_2, \ldots, c_n$.

- *Initialization*: each variable node x_k sends to all its check nodes the message $f_k^{(0)}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k - x)^2}{2\sigma^2}}$.
- Basic iteration – check node message: Each check node sends a (different) message to each of the variable nodes that are connected to it. For a specific check node denote (without loss of generality) the appropriate check equation by $\sum_{l=1}^{r} h_l x_{m_l} = integer$, where $x_{m_l}, l = 1, 2, \ldots, r$ are the variable nodes that are connected to this check node (and r is the appropriate row degree of $\boldsymbol{H}$). Denote by $f_l(x)$, $l = 1, 2, \ldots, r$, the message that was sent to this check node by variable node x_{m_l} in the previous half-iteration. The message that the check node transmits back to variable node x_{m_j} is calculated in three basic steps.
  1) The convolution step – all messages, except $f_j(x)$, are convolved (after expanding each $f_l(x)$ by h_l):

$$\tilde{p}_j(x) = f_1\left(\frac{x}{h_1}\right) * \cdots f_{j-1}\left(\frac{x}{h_{j-1}}\right) * f_{j+1}\left(\frac{x}{h_{j+1}}\right) * \cdots\cdots * f_r\left(\frac{x}{h_r}\right). \quad (2)$$

  2) The stretching step—The result is stretched by $(-h_j)$ to $p_j(x) = \tilde{p}_j(-h_j x)$
  3) The periodic extension step—The result is extended to a periodic function with period $1/|h_j|$:

$$Q_j(x) = \sum_{i=-\infty}^{\infty} p_j\left(x - \frac{i}{h_j}\right). \quad (3)$$

  The function $Q_j(x)$ is the message that is finally sent to variable node x_{m_j}.
- Basic iteration—variable node message: Each variable node sends a (different) message to each of the check nodes that are connected to it. For a specific variable node x_k, assume that it is connected to check nodes $c_{m_1}, c_{m_2}, \ldots, c_{m_e}$, where e is the appropriate column degree of $\boldsymbol{H}$. Denote by $Q_l(x)$, $l = 1, 2, \ldots, e$, the message that was sent from check node c_{m_l} to this variable node in the previous half-iteration. The message that is sent back to check node c_{m_j} is calculated in two basic steps:
  1) the product step

$$\tilde{f}_j(x) = e^{-\frac{(y_k - x)^2}{2\sigma^2}} \prod_{\substack{l=1 \\ l \neq j}}^{e} Q_l(x);$$

  2) the normalization step:

$$f_j(x) = \frac{\tilde{f}_j(x)}{\int_{-\infty}^{\infty} \tilde{f}_j(x) dx}.$$

This basic iteration is repeated for the desired number of iterations.

- *Final decision*: After finishing the iterations, we want to estimate the integer information vector $\underline{b}$. First, we estimate the final pdfs of the codeword elements x_k, $k = 1, 2, \ldots, n$, by calculating the variable node messages at the last iteration without omitting any check node message in the product step

$$\tilde{f}^{(k)}_{final}(x) = e^{-\frac{(y_k - x)^2}{2\sigma^2}} \prod_{l=1}^{e} Q_l(x).$$

.Then, we estimate each x_k by finding the peak of its pdf: $\hat{x_k} = arg\max_x \tilde{f}^{(k)}_{final}(x)$. Finally, we estimate $\underline{b}$ as $\hat{\underline{b}} = \lfloor \boldsymbol{H}\hat{\underline{x}} \rceil$. See Appendix III for a possible simplification of the calculations at the last iteration, and for an alternative method to estimate $\underline{b}$.

Instead of developing the algorithm using Gallager's principles, we can use the sum-product algorithm [23], used in graphical models [24], for the bipartite graph of the LDLC code, and get the same decoding procedure. Specifically, assuming that all dependencies are coming from the constraints represented by the graph, the check node operation is equivalent to calculating the pdf of x_{m_j} from the pdfs of x_{m_i}, $i = 1, 2, \ldots, j-1, j+1, \ldots, r$, given that $\sum_{l=1}^{r} h_l x_{m_l} = integer$, and assuming that x_{m_i} are independent. Extracting x_{m_j} from the check equation, we get

$$x_{m_j} = \frac{1}{h_j}\left(integer - \sum_{\substack{l=1 \\ l \neq j}}^{r} h_l x_{m_l} \right).$$

Since the pdf of a sum of independent random variables is the convolution of the corresponding pdfs, (2) and the stretching step that follows it simply calculate the pdf of x_{m_j}, assuming that the integer at the right hand side of the check equation is zero. The result is then periodically extended such that a properly shifted copy exists for every possible value of this (unknown) integer. The variable node gets such a message from all the check equations that involve the corresponding variable. The check node messages and the channel pdf are treated as independent sources of information on the variable, so they are multiplied all together.

Note that the periodic extension step at the check nodes is equivalent to a convolution with an infinite impulse train. With this observation, the operation of the variable nodes is completely analogous to that of the check nodes: the variable nodes multiply the incoming messages by the channel pdf, whereas the check nodes convolve the incoming messages with an impulse train, which can be regarded as a generalized "integer pdf".

In the above formulation, the integer information vector $\underline{b}$ is recovered from the pdfs of the codeword elements x_k. An alternative approach is to calculate the pdf of each integer element b_m directly as the pdf of the left hand side of the appropriate check equation. Using the tree assumption, this can be done by simply calculating the convolution $\tilde{p}(x)$ as in (2), but this time without omitting any pdf, i.e., all the received variable node messages are convolved. Then, the integer b_m is determined by $\hat{b}_m = arg\max_{j \in \mathbb{Z}} \tilde{p}(j)$.

Fig. 3 shows an example for a regular LDLC with degree $d = 5$. The figure shows all the signals that are involved in generating a variable node message for a certain variable node. The top signal is the channel Gaussian, centered around the noisy observation of the variable. The next four signals are the periodically extended pdfs that arrived from the check nodes, and the bottom signal is the product of all the five signals. It can be seen that each periodic signal has a different period, according to the relevant coefficient of $\boldsymbol{H}$. Also, the signals with larger period have larger variance. This diversity resolves all the ambiguities such that the multiplication result (bottom plot) remains with a single peak. We expect the iterative algorithm to converge to a solution where a single peak will remain at each pdf, located at the desired value and narrow enough to estimate the information.

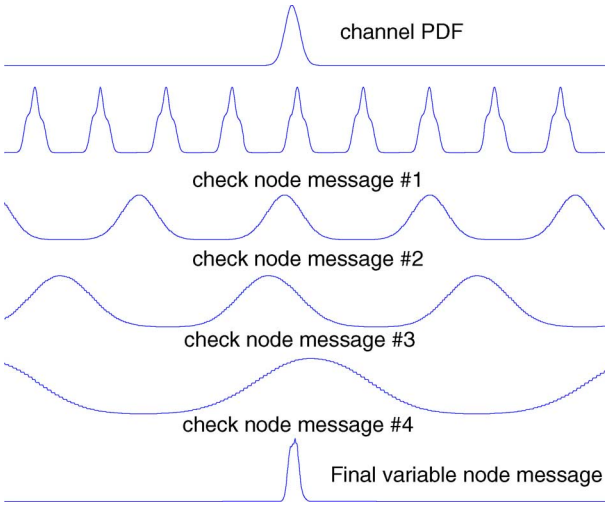

Fig. 3. Signals at variable node.

IV. Convergence

A. The Gaussian Mixture Model

Interestingly, for LDLC we can come up with a convergence analysis that in many respects is more specific than the similar analysis for LDPC codes.

We start by introducing basic claims about Gaussian pdfs. Denote $G_{m,V}(x) = \frac{1}{\sqrt{2\pi V}} e^{-\frac{(x-m)^2}{2V}}$.

Claim 1 (Convolution of Gaussians): The convolution of n Gaussians with mean values $m_1, m_2, \ldots, m_n$ and variances $V_1, V_2, \ldots, V_n$, respectively, is a Gaussian with mean $m_1 + m_2 + \cdots + m_n$ and variance $V_1 + V_2 + \cdots + V_n$.

Proof: See [25]. □

Claim 2 (Product of n Gaussians): Let

$$G_{m_1,V_1}(x), G_{m_2,V_2}(x), \ldots, G_{m_n,V_n}(x)$$

be n Gaussians with mean values $m_1, m_2, \ldots, m_n$ and variances $V_1, V_2, \ldots, V_n$ respectively. Then, the product of these Gaussians is a scaled Gaussian:

$$\prod_{i=1}^{n} G_{m_i,V_i}(x) = \hat{A} \cdot G_{\hat{m},\hat{V}}(x)$$

where $\frac{1}{\hat{V}} = \sum_{i=1}^{n} \frac{1}{V_i}$, $\hat{m} = \frac{\sum_{i=1}^{n} m_i V_i^{-1}}{\sum_{i=1}^{n} V_i^{-1}}$, and

$$\hat{A} = \frac{1}{\sqrt{(2\pi)^{n-1} \hat{V}^{-1} \prod_{k=1}^{n} V_k}} \cdot e^{-\frac{\hat{m}^2}{2\hat{V}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(m_i - m_j)^2}{V_i \cdot V_j}}.$$

Proof: By straightforward mathematical manipulations. □

The reason that we are interested in the properties of Gaussian pdfs lies in the following lemma.

Lemma 1: Each message that is exchanged between the check nodes and variable nodes in the LDLC decoding algorithm (i.e., $Q_j(x)$ and $f_j(x)$), at every iteration, can be expressed as a Gaussian mixture of the form $M(x)=\sum_{j=1}^{\infty} A_j G_{m_j,V_j}(x)$.

Proof: By induction: The initial messages are Gaussians, and the basic operations of the iterative decoder preserve the Gaussian mixture nature of Gaussian mixture inputs (convolution and multiplication preserve the Gaussian nature according to claims 1 and 2, stretching, expanding and shifting preserve it by the definition of a Gaussian, and periodic extension transforms a single Gaussian to a mixture and a mixture to a mixture). □

Convergence analysis should therefore analyze the convergence of the variances, mean values and amplitudes of the Gaussians in each mixture.

B. Convergence of the Variances

We shall now analyze the behavior of the variances, and start with the following lemma.

Lemma 2: For both variable node messages and check node messages, all the Gaussians that take place in the same mixture have the same variance.

Proof: By induction. The initial variable node messages are single element mixtures so the claim obviously holds. Assume now that all the variable node messages at iteration t are mixtures where all the Gaussians that take place in the same mixture have the same variance. In the convolution step (2), each variable node message is first expanded. All Gaussians in the expanded mixture will still have the same variance, since the whole mixture is expanded together. Then, $d-1$ expanded Gaussian mixtures are convolved. In the resulting mixture, each Gaussian will be the result of convolving $d-1$ single Gaussians, one from each mixture. According to claim 1, all the Gaussians in the convolution result will have the same variance, which will equal the sum of the $d-1$ variances of the expanded messages. The stretching and periodic extension (3) do not change the equal variance property, so it holds for the final check node messages. The variable nodes multiply $d-1$ check node messages. Each Gaussian in the resulting mixture is a product of $d-1$ single Gaussians, one from each mixture, and the channel noise Gaussian. According to claim 2, they will all have the same variance. The final normalization step does not change the variances so the equal variance property is kept for the final variable node messages at iteration $t+1$. □

Until this point we did not impose any restrictions on the LDLC. From now on, we shall restrict ourselves to Latin square regular LDLC (see Definition 3). The basic iterative equations that relate the variances at iteration $t+1$ to the variances at iteration t are summarized in the following two lemmas.

Lemma 3: For Latin square LDLC, variable node messages that are sent at the same iteration along edges with the same absolute value have the same variance.

Proof: See Appendix IV. □

Lemma 4: For Latin square LDLC with degree d, denote the variance of the messages that are sent at iteration t along edges with weight $\pm h_l$ by $V_l^{(t)}$. The variance values $V_1^{(t)}, V_2^{(t)}, \ldots, V_d^{(t)}$ obey the following recursion:

$$\frac{1}{V_i^{(t+1)}} = \frac{1}{\sigma^2} + \sum_{\substack{m=1 \\ m\neq i}}^{d} \frac{h_m^2}{\sum_{\substack{j=1 \\ j\neq m}}^{d} h_j^2 V_j^{(t)}} \tag{4}$$

for $i=1,2,\ldots,d$, with initial conditions $V_1^{(0)}=V_2^{(0)}=\cdots=V_d^{(0)}=\sigma^2$.

Proof: See Appendix IV. □

For illustration, the recursion for the case $d=3$ is

$$\begin{aligned}
\frac{1}{V_1^{(t+1)}} &= \frac{h_2^2}{h_1^2 V_1^{(t)} + h_3^2 V_3^{(t)}} + \frac{h_3^2}{h_1^2 V_1^{(t)} + h_2^2 V_2^{(t)}} + \frac{1}{\sigma^2} \\
\frac{1}{V_2^{(t+1)}} &= \frac{h_1^2}{h_2^2 V_2^{(t)} + h_3^2 V_3^{(t)}} + \frac{h_3^2}{h_1^2 V_1^{(t)} + h_2^2 V_2^{(t)}} + \frac{1}{\sigma^2} \\
\frac{1}{V_3^{(t+1)}} &= \frac{h_1^2}{h_2^2 V_2^{(t)} + h_3^2 V_3^{(t)}} + \frac{h_2^2}{h_1^2 V_1^{(t)} + h_3^2 V_3^{(t)}} + \frac{1}{\sigma^2}.
\end{aligned} \tag{5}$$

The lemmas above are used to prove the following theorem regarding the convergence of the variances.

Theorem 1: For a Latin square LDLC with degree d and generating sequence $h_1 \geq h_2 \geq \cdots \geq h_d > 0$, define $\alpha \triangleq \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$. Assume that $\alpha < 1$. Then we have the following.

1) The first variance approaches a constant value of $\sigma^2(1-\alpha)$, where σ^2 is the channel noise variance
$$V_1^{(\infty)} \triangleq \lim_{t\to\infty} V_1^{(t)} = \sigma^2(1-\alpha).$$
2) The other variances approach zero
$$V_i^{(\infty)} \triangleq \lim_{t\to\infty} V_i^{(t)} = 0$$
for $i=2,3,\ldots,d$.
3) The asymptotic convergence rate of all variances is exponential
$$0 < \lim_{t\to\infty} \left| \frac{V_i^{(t)} - V_i^{(\infty)}}{\alpha^t} \right| < \infty$$
for $i=1,2,\ldots,d$.
4) The zero approaching variances are upper bounded by the decaying exponential $\sigma^2\alpha^t$
$$V_i^{(t)} \leq \sigma^2 \alpha^t$$
for $i=2,3,\ldots,d$ and $t \geq 0$.

Proof: See Appendix IV. □

If $\alpha \geq 1$, the variances may still converge, but convergence rate may be as slow as $o(1/t)$, as illustrated in Appendix IV.

Convergence of the variances to zero implies that the Gaussians approach impulses. This is a desired property of the decoder, since the exact pdf that we want to calculate is indeed a

weighted sum of impulses (see (1)). It can be seen that by designing a code with $\alpha < 1$, i.e., $h_1^2 > \sum_{i=2}^{d} h_i^2$, one variance approaches a constant (and not zero). However, all the other variances approach zero, where all variances converge in an exponential rate. This will be the preferred mode because the information can be recovered even if a single variance does not decay to zero, where exponential convergence is certainly preferred over slow $1/t$ convergence. Therefore, from now on we shall restrict our analysis to Latin square LDLC with $\alpha < 1$.

Theorem 1 shows that every iteration, each variable node will generate $d-1$ messages with variances that approach zero, and a single message with variance that approaches a constant. The message with nonzero variance will be transmitted along the edge with largest weight (i.e., h_1). However, from the derivation of Appendix IV it can be seen that the opposite happens for the check nodes: each check node will generate $d-1$ messages with variances that approach a constant, and a single message with variance that approaches zero. The check node message with zero approaching variance will be transmitted along the edge with largest weight.

C. Convergence of the Mean Values

The reason that the messages are mixtures and not single Gaussians lies in the periodic extension step (3) at the check nodes, and every Gaussian at the output of this step can be related to a single index of the infinite sum. Therefore, we can label each Gaussian at iteration t with a list of all the indices that were used in (3) during its creation process in iterations $1, 2, \ldots, t$.

Definition 4 (Label of a Gaussian): The label of a Gaussian consists of a sequence of triplets of the form $\{t, c, i\}$, where t is an iteration index, c is a check node index and i is an integer. The labels are initialized to the empty sequence. Then, the labels are updated along each iteration according to the following update rules.

1) In the periodic extension step (3), each Gaussian in the output periodic mixture is assigned the label of the specific Gaussian of $p_j(x)$ that generated it, concatenated with a single triplet $\{t, c, i\}$, where t is the current iteration index, c is the check node index and i is the index in the infinite sum of (3) that corresponds to this Gaussian.
2) In the convolution step and the product step, each Gaussian in the output mixture is assigned a label that equals the concatenation of all the labels of the specific Gaussians in the input messages that formed this Gaussian.
3) The stretching and normalization steps do not alter the label of each Gaussian: Each Gaussian in the stretched/normalized mixture inherits the label of the appropriate Gaussian in the original mixture.

Definition 5 (A Consistent Gaussian): A Gaussian in a mixture is called "$[t_a, t_b]$ consistent" if its label contains no contradictions for iterations t_a to t_b, i.e., for every pair of triplets $\{t_1, c_1, i_1\}$, $\{t_2, c_2, i_2\}$ such that $t_a \le t_1, t_2 \le t_b$, if $c_1 = c_2$ then $i_1 = i_2$. A $[0, \infty]$ consistent Gaussian will be simply called a consistent Gaussian.

We can relate every consistent Gaussian to a unique integer vector $\underline{b} \in \mathbb{Z}^n$, which holds the n integers used in the n check nodes. Since in the periodic extension step (3) the sum is taken over all integers, a consistent Gaussian exists in each variable node message for every possible integer valued vector $\underline{b} \in \mathbb{Z}^n$. We shall see later that this consistent Gaussian corresponds to the lattice point $\boldsymbol{G}\underline{b}$.

According to Theorem 1, if we choose the nonzero values of $\boldsymbol{H}$ such that $\alpha < 1$, every variable node generates $d-1$ messages with variances approaching zero and a single message with variance that approaches a constant. We shall refer to these messages as "narrow" messages and "wide" messages, respectively. For a given integer valued vector $\underline{b}$, we shall concentrate on the consistent Gaussians that relate to $\underline{b}$ in all the nd variable node messages that are generated in each iteration (a single Gaussian in each message). The following lemmas summarize the asymptotic behavior of the mean values of these consistent Gaussians for the narrow messages.

Lemma 5: For a Latin square LDLC with degree d and $\alpha < 1$, consider the $d-1$ narrow messages that are sent from a specific variable node. Consider further a single Gaussian in each message, which is the consistent Gaussian that relates to a given integer vector $\underline{b}$. Asymptotically, the mean values of these $d-1$ Gaussians become equal.

Proof: See Appendix V. □

Lemma 6: For a Latin square LDLC with dimension n, degree d and $\alpha < 1$, consider only consistent Gaussians that relate to a given integer vector $\underline{b}$ and belong to narrow messages. Denote the common mean value of the $d-1$ such Gaussians that are sent from variable node i at iteration t by $m_i^{(t)}$, and arrange all these mean values in a column vector $\underline{m}^{(t)}$ of dimension n. Define the error vector $\underline{e}^{(t)} \triangleq \underline{m}^{(t)} - \underline{x}$, where $\underline{x} = \boldsymbol{G}\underline{b}$ is the lattice point that corresponds to $\underline{b}$. Then, for large t, $\underline{e}^{(t)}$ satisfies

$$\underline{e}^{(t+1)} \approx -\tilde{\boldsymbol{H}} \cdot \underline{e}^{(t)} \tag{6}$$

where $\tilde{\boldsymbol{H}}$ is derived from $\boldsymbol{H}$ by permuting the rows such that the $\pm h_1$ elements will be placed on the diagonal, dividing each row by the appropriate diagonal element (h_1 or $-h_1$), and then nullifying the diagonal.

Proof: See Appendix V. □

We can now state the following theorem, which describes the conditions for convergence and the steady-state value of the mean values of the consistent Gaussians of the narrow variable node messages.

Theorem 2: For a Latin square LDLC with $\alpha < 1$, the mean values of the consistent Gaussians of the narrow variable node messages that relate to a given integer vector $\underline{b}$ are assured to converge if and only if all the eigenvalues of $\tilde{\boldsymbol{H}}$ have magnitude less than 1, where $\tilde{\boldsymbol{H}}$ is defined in Lemma 6. When this condition is fulfilled, the mean values converge to the coordinates of the appropriate lattice point: $\underline{m}^{(\infty)} = \boldsymbol{G} \cdot \underline{b}$.

Proof: Immediate from Lemma 6. □

Note that without adding random signs to the LDLC nonzero values, the all-ones vector will be an eigenvector of $\tilde{\boldsymbol{H}}$ with eigenvalue $\frac{\sum_{i=2}^{d} h_i}{h_1}$, which may exceed 1.

Interestingly, recursion (6) is also obeyed by the error of the Jacobi method for solving systems of sparse linear equations [26] (see also Section VIII-A), when it is used to solve $\boldsymbol{H}\underline{m} = \underline{b}$ (with solution $\underline{m} = \boldsymbol{G}\underline{b}$). Therefore, the LDLC decoder can be viewed as a superposition of Jacobi solvers, one for each possible value of the integer valued vector $\underline{b}$.

We shall now turn to the convergence of the mean values of the wide messages. The asymptotic behavior is summarized in the following lemma.

Lemma 7: For a Latin square LDLC with dimension n and $\alpha < 1$, consider only consistent Gaussians that relate to a given integer vector $\underline{b}$ and belong to wide messages. Denote the mean value of such a Gaussian that is sent from variable node i at iteration t by $m_i^{(t)}$, and arrange all these mean values in a column vector $\underline{m}^{(t)}$ of dimension n. Define the error vector $\underline{e}^{(t)} \triangleq \underline{m}^{(t)} - \boldsymbol{G}\underline{b}$. Then, for large t, $\underline{e}^{(t)}$ satisfies

$$\underline{e}^{(t+1)} \approx -\boldsymbol{F} \cdot \underline{e}^{(t)} + (1-\alpha)(\underline{y} - \boldsymbol{G}\underline{b}) \tag{7}$$

where $\underline{y}$ is the noisy codeword and $\boldsymbol{F}$ is an $n \times n$ matrix defined by

$$F_{k,l} = \begin{cases} \frac{H_{r,k}}{H_{r,l}}, & \text{if } k \neq l \text{ and there exist a row } r \text{ of } H \\ & \text{for which } |H_{r,l}| = h_1 \text{ and } H_{r,k} \neq 0 \\ 0, & \text{otherwise} \end{cases} . \tag{8}$$

Proof: See Appendix V, where an alternative way to construct $\boldsymbol{F}$ from $\boldsymbol{H}$ is also presented. □

The conditions for convergence and steady-state solution for the wide messages are described in the following theorem.

Theorem 3: For a Latin square LDLC with $\alpha < 1$, the mean values of the consistent Gaussians of the wide variable node messages that relate to a given integer vector $\underline{b}$ are assured to converge if and only if all the eigenvalues of $\boldsymbol{F}$ have magnitude less than 1, where $\boldsymbol{F}$ is defined in Lemma 7. When this condition is fulfilled, the steady-state solution is

$$\underline{m}^{(\infty)} = \boldsymbol{G} \cdot \underline{b} + (1-\alpha)(\boldsymbol{I} + \boldsymbol{F})^{-1}(\underline{y} - \boldsymbol{G} \cdot \underline{b}).$$

Proof: Immediate from Lemma 7. □

Unlike the narrow messages, the mean values of the wide messages do not converge to the appropriate lattice point coordinates. The steady-state error depends on the difference between the noisy observation and the lattice point, as well as on α, and it decreases to zero as $\alpha \to 1$. Note that the final pdf of a variable is generated by multiplying *all* the d check node messages that arrive to the appropriate variable node. $d-1$ of these messages are wide, and therefore their mean values have a steady-state error. One message is narrow, so it converges to an impulse at the lattice point coordinate. Therefore, the final product will be an impulse at the correct location, where the wide messages will only affect the magnitude of this impulse. As long as the mean values errors are not too large (relative to the width of the wide messages), this should not cause an impulse that corresponds to a wrong lattice point to have larger amplitude than the correct one. However, for large noise, these steady-state errors may cause the decoder to deviate from the ML solution (As explained in Section IV-D).

To summarize the results for the mean values, we considered the mean values of all the consistent Gaussians that correspond to a given integer vector $\underline{b}$. A single Gaussian of this form exists in each of the nd variable node messages that are generated in each iteration. For each variable node, $d-1$ messages are narrow (have variance that approaches zero) and a single message is wide (variance approaches a constant). Under certain conditions on $\boldsymbol{H}$, the mean values of all the narrow messages converge to the appropriate coordinate of the lattice point $\boldsymbol{G}\underline{b}$. Under additional conditions on $\boldsymbol{H}$, the mean values of the wide messages converge, but the steady-state values contain an error term.

We analyzed the behavior of consistent Gaussian. It should be noted that there are many more nonconsistent Gaussians. Furthermore, nonconsistent Gaussians are generated in each iteration for any existing consistent Gaussian. We conjecture that unless a Gaussian is consistent, or becomes consistent along the iterations, it fades out, at least at noise conditions where the algorithm converges. The reason is that nonconsistency in the integer values leads to mismatch in the corresponding pdfs, and so the amplitude of that Gaussian is attenuated.

We considered consistent Gaussians which correspond to a specific integer vector $\underline{b}$, but such a set of Gaussians exists for every possible choice of $\underline{b}$, i.e., for every lattice point. Therefore, the narrow messages will converge to a solution that has an impulse at the appropriate coordinate of every lattice point. This resembles the exact solution (1), so the key for proper convergence lies in the amplitudes: we would like the consistent Gaussians of the ML lattice point to have the largest amplitude for each message.

D. Convergence of the Amplitudes

We shall now analyze the behavior of the amplitudes of consistent Gaussians (as discussed later, this is not enough for complete convergence analysis, but it certainly gives insight to the nature of the convergence process and its properties). The behavior of the amplitudes of consistent Gaussians is summarized in the following lemma.

Lemma 8: For a Latin square LDLC with dimension n, degree d and $\alpha < 1$, consider the nd consistent Gaussians that relate to a given integer vector $\underline{b}$ in the variable node messages that are sent at iteration t (one consistent Gaussian per message). Denote the amplitudes of these Gaussians by $p_i^{(t)}, i = 1, 2, \ldots, nd$, and define the log-amplitude as $l_i^{(t)} = \log p_i^{(t)}$. Arrange these nd log-amplitudes in a column vector $\underline{l}^{(t)}$, such that element $(k-1)d + i$ corresponds to the message that is sent from variable node k along an edge with weight $\pm h_i$. Assume further that the bipartite graph of the LDLC contains no 4-loops. Then, the log-amplitudes satisfy the following recursion:

$$\underline{l}^{(t+1)} = \boldsymbol{A} \cdot \underline{l}^{(t)} - \underline{a}^{(t)} - \underline{c}^{(t)} \tag{9}$$

with initialization $\underline{l}^{(0)} = \underline{0}$. $\boldsymbol{A}$ is an $nd \times nd$ matrix which is all zeros except for exactly $(d-1)^2$ "1"s in each row and each column. The element of the excitation vector $\underline{a}^{(t)}$ at location $(k-1)d+i$ (where $k = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, d$) equals

$$a^{(t)}_{(k-1)d+i} = \frac{\hat{V}^{(t)}_{k,i}}{2}\left(\sum_{\substack{l=1\\ l\neq i}}^{d}\sum_{\substack{j=l+1\\ j\neq i}}^{d}\frac{\left(\tilde{m}^{(t)}_{k,l}-\tilde{m}^{(t)}_{k,j}\right)^2}{\tilde{V}^{(t)}_{k,l}\cdot\tilde{V}^{(t)}_{k,j}}+\sum_{\substack{l=1\\ l\neq i}}^{d}\frac{\left(\tilde{m}^{(t)}_{k,l}-y_k\right)^2}{\sigma^2\cdot\tilde{V}^{(t)}_{k,l}}\right) \quad (10)$$

where $\tilde{m}^{(t)}_{k,l}$ and $\tilde{V}^{(t)}_{k,l}$ denote the mean value and variance of the consistent Gaussian that relates to the integer vector $\underline{b}$ in the check node message that arrives to variable node k at iteration t along an edge with weight $\pm h_l$. y_k is the noisy channel observation of variable node k, and

$$\hat{V}^{(t)}_{k,i} \triangleq \left(\frac{1}{\sigma^2}+\sum_{\substack{l=1\\ l\neq i}}^{d}\frac{1}{\tilde{V}^{(t)}_{k,l}}\right)^{-1}.$$

Finally, $\underline{c}^{(t)}$ is a constant excitation term that is independent of the integer vector $\underline{b}$ (i.e., is the same for all consistent Gaussians). Note that an iteration is defined as sending variable node messages, followed by sending check node messages. The first iteration (where the variable nodes send the initialization pdf) is regarded as iteration 0.

Proof: At the check node, the amplitude of a Gaussian at the convolution output is the product of the amplitudes of the corresponding Gaussians in the appropriate variable node messages. At the variable node, the amplitude of a Gaussian at the product output is the product of the amplitudes of the corresponding Gaussians in the appropriate check node messages, multiplied by the Gaussian scaling term, according to claim 2. Since we assume that the bipartite graph of the LDLC contains no 4-loops, an amplitude of a variable node message at iteration t will therefore equal the product of $(d-1)^2$ amplitudes of Gaussians of variable node messages from iteration $t-1$, multiplied by the Gaussian scaling term. This proves (9) and shows that A has $(d-1)^2$ "1"s in every row. However, since each variable node message affects exactly $(d-1)^2$ variable node messages of the next iteration, A must also have $(d-1)^2$ "1"s in every column. The total excitation term $-\underline{a}^{(t)}-\underline{c}^{(t)}$ corresponds to the logarithm of the Gaussian scaling term. Each element of this scaling term results from the product of $d-1$ check node Gaussians and the channel Gaussian, according to claim 2. This scaling term sums over all the pairs of Gaussians, and in (10) the sum is separated to pairs that include the channel Gaussian and pairs that do not. The total excitation is divided between (10), which depends on the choice of the integer vector $\underline{b}$, and $\underline{c}^{(t)}$, which includes all the constant terms that are independent on $\underline{b}$ (including the normalization operation which is performed at the variable node). □

Since there are exactly $(d-1)^2$ "1"s in each column of the matrix $\boldsymbol{A}$, it is easy to see that the all-ones vector is an eigenvector of $\boldsymbol{A}$, with eigenvalue $(d-1)^2$. If $d > 2$, this eigenvalue is larger than 1, meaning that the recursion (9) is nonstable.

It can be seen that the excitation term $\underline{a}^{(t)}$ has two components. The first term sums the squared differences between the mean values of all the possible pairs of received check node messages (weighted by the inverse product of the appropriate variances). It therefore measures the mismatch between the incoming messages. This mismatch will be small if the mean values of the consistent Gaussians converge to the coordinates of a lattice point (*any* lattice point). The second term sums the squared differences between the mean values of the incoming messages and the noisy channel output y_k. This term measures the mismatch between the incoming messages and the channel measurement. It will be smallest if the mean values of the consistent Gaussians converge to the coordinates of the ML lattice point.

The following lemma summarizes some properties of the excitation term $\underline{a}^{(t)}$.

Lemma 9: For a Latin square LDLC with dimension n, degree d, $\alpha < 1$ and no 4-loops, consider the consistent Gaussians that correspond to a given integer vector $\underline{b}$. According to Lemma 8, their amplitudes satisfy recursion (9). The excitation term $\underline{a}^{(t)}$ of (9), which is defined by (10), satisfies the following properties:

1) $a^{(t)}_i$, the ith element of $\underline{a}^{(t)}$, is nonnegative, finite and bounded for every i and every t. Moreover, $a^{(t)}_i$ converges to a finite nonnegative steady-state value as $t \to \infty$.
2) $\lim_{t\to\infty}\sum_{i=1}^{nd} a^{(t)}_i = \frac{1}{2\sigma^2}(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$, where $\underline{y}$ is the noisy received codeword and $\boldsymbol{W}$ is a positive definite matrix defined by

$$\boldsymbol{W} \triangleq (d+1-\alpha)\boldsymbol{I} - 2(1-\alpha)(\boldsymbol{I}+\boldsymbol{F})^{-1} + \\ +(1-\alpha)(\boldsymbol{I}+\boldsymbol{F})^{-1^T}\left((d-1)^2\boldsymbol{I}-\boldsymbol{F}^T\boldsymbol{F}\right)(\boldsymbol{I}+\boldsymbol{F})^{-1} \quad (11)$$

   where $\boldsymbol{F}$ is defined in Lemma 7.
3) For an LDLC with degree $d > 2$, the weighted infinite sum $\sum_{j=0}^{\infty}\frac{\sum_{i=1}^{nd} a_i^{(j)}}{(d-1)^{2j+2}}$ converges to a finite value.

Proof: See Appendix VI. □

The following theorem addresses the question of which consistent Gaussian will have the maximal asymptotic amplitude. We shall first consider the case of an LDLC with degree $d > 2$, and then consider the special case of $d = 2$ in a separate theorem.

Theorem 4: For a Latin square LDLC with dimension n, degree $d > 2$, $\alpha < 1$ and no 4-loops, consider the nd consistent Gaussians that relate to a given integer vector $\underline{b}$ in the variable node messages that are sent at iteration t (one consistent Gaussian per message). Denote the amplitudes of these Gaussians by $p^{(t)}_i$, $i = 1, 2, \ldots, nd$, and define the product-of-amplitudes as $P^{(t)} \triangleq \prod_{i=1}^{nd} p^{(t)}_i$. Define further $S = \sum_{j=0}^{\infty}\frac{\sum_{i=1}^{nd} a_i^{(j)}}{(d-1)^{2j+2}}$, where $a^{(j)}_i$ is defined by (10) (S is well defined according to Lemma 9). Then, we have the following.

1) The integer vector $\underline{b}$ for which the consistent Gaussians will have the largest asymptotic product-of-amplitudes $\lim_{t\to\infty} P^{(t)}$ is the one for which S is minimized.

2) The product-of-amplitudes for the consistent Gaussians that correspond to all other integer vectors will decay to zero in a super-exponential rate.

Proof: As in Lemma 8, define the log-amplitudes $l_i^{(t)} \triangleq \log p_i^{(t)}$. Define further $s^{(t)} \triangleq \sum_{i=1}^{nd} l_i^{(t)}$. Taking the element-wise sum of (9), we get:

$$s^{(t+1)} = (d-1)^2 s^{(t)} - \sum_{i=1}^{nd} a_i^{(t)} \tag{12}$$

with initialization $s^{(0)} = 0$. Note that we ignored the term $\sum_{i=1}^{nd} c_i^{(t)}$. As shown below, we are looking for the vector $\underline{b}$ that maximizes $s^{(t)}$. Since (12) is a linear difference equation, and the term $\sum_{i=1}^{nd} c_i^{(t)}$ is independent of $\underline{b}$, its effect on $s^{(t)}$ is common to all $\underline{b}$ and is therefore not interesting.

Define now $\tilde{s}^{(t)} \triangleq \frac{s^{(t)}}{(d-1)^{2t}}$. Substituting in (12), we get

$$\tilde{s}^{(t+1)} = \tilde{s}^{(t)} - \frac{1}{(d-1)^{2t+2}} \sum_{i=1}^{nd} a_i^{(t)} \tag{13}$$

with initialization $\tilde{s}^{(0)} = 0$, which can be solved to get

$$\tilde{s}^{(t)} = -\sum_{j=0}^{t-1} \frac{\sum_{i=1}^{nd} a_i^{(j)}}{(d-1)^{2j+2}}. \tag{14}$$

We would now like to compare the amplitudes of consistent Gaussians with various values of the corresponding integer vector $\underline{b}$ in order to find the lattice point whose consistent Gaussians will have largest product-of-amplitudes. From the definitions of $s^{(t)}$ and $\tilde{s}^{(t)}$ we then have

$$P^{(t)} = e^{s^{(t)}} = e^{(d-1)^{2t} \cdot \tilde{s}^{(t)}}. \tag{15}$$

Consider two integer vectors $\underline{b}$ that relate to two lattice points. Denote the corresponding product-of-amplitudes by $P_0^{(t)}$ and $P_1^{(t)}$, respectively, and assume that for these two vectors S converges to the values S_0 and S_1, respectively. Then, taking into account that $\lim_{t\to\infty} \tilde{s}^{(t)} = -S$, the asymptotic ratio of the product-of-amplitudes for these lattice points will be:

$$\lim_{t\to\infty} \frac{P_1^{(t)}}{P_0^{(t)}} = \frac{e^{-(d-1)^{2t} \cdot S_1}}{e^{-(d-1)^{2t} \cdot S_0}} = e^{(d-1)^{2t} \cdot (S_0 - S_1)}. \tag{16}$$

It can be seen that if $S_0 < S_1$, the ratio decreases to zero in a super exponential rate. This shows that the lattice point for which S is minimized will have the largest product-of-amplitudes, where for all other lattice points, the product-of-amplitudes will decay to zero in a super-exponential rate (recall that the normalization operation at the variable node keeps the sum of all amplitudes in a message to be 1). This completes the proof of the theorem. □

We now have to find which integer valued vector $\underline{b}$ minimizes S. The analysis is difficult because the weighting factor inside the sum of (14) performs exponential weighting of the excitation terms, where the dominant terms are those of the first iterations. Therefore, we cannot use the asymptotic results of Lemma 9, but have to analyze the transient behavior. However, the analysis is simpler for the case of an LDLC with row and column degree of $d = 2$, so we shall first turn to this simple case (note that for this case, both the convolution in the check nodes and the product at the variable nodes involve only a single message).

Theorem 5: For a Latin square LDLC with dimension n, degree $d = 2$, $\alpha < 1$ and no 4-loops, consider the $2n$ consistent Gaussians that relate to a given integer vector $\underline{b}$ in the variable node messages that are sent at iteration t (one consistent Gaussian per message). Denote the amplitudes of these Gaussians by $p_i^{(t)}, i = 1, 2, \ldots, 2n$, and define the product-of-amplitudes as $P^{(t)} \triangleq \prod_{i=1}^{2n} p_i^{(t)}$. Then:

1) The integer vector $\underline{b}$ for which the consistent Gaussians will have the largest asymptotic product-of-amplitudes $\lim_{t\to\infty} P^{(t)}$ is the one for which $(\boldsymbol{G}\underline{b} - \underline{y})^T \boldsymbol{W} (\boldsymbol{G}\underline{b} - \underline{y})$ is minimized, where $\boldsymbol{W}$ is defined by (11) and $\underline{y}$ is the noisy received codeword.
2) The product-of-amplitudes for the consistent Gaussians that correspond to all other integer vectors will decay to zero in an exponential rate.

Proof: For $d = 2$ (12) becomes

$$s^{(t+1)} = s^{(t)} - \sum_{i=1}^{2n} a_i^{(t)}. \tag{17}$$

With solution:

$$s^{(t)} = -\sum_{j=0}^{t-1} \sum_{i=1}^{2n} a_i^{(j)}. \tag{18}$$

Denote $S_a = \lim_{j\to\infty} \sum_{i=1}^{2n} a_i^{(j)}$. S_a is well defined according to Lemma 9. For large t, we then have $s^{(t)} \approx -t \cdot S_a$. Therefore, for two lattice points with excitation sum terms which approach S_{a0}, S_{a1}, respectively, the ratio of the corresponding product-of-amplitudes will approach

$$\lim_{t\to\infty} \frac{P_1^{(t)}}{P_0^{(t)}} = \frac{e^{-S_{a1} \cdot t}}{e^{-S_{a0} \cdot t}} = e^{(S_{a0} - S_{a1}) \cdot t}. \tag{19}$$

If $S_{a0} < S_{a1}$, the ratio decreases to zero exponentially (unlike the case of $d > 2$ where the rate was super-exponential, as in (16)). This shows that the lattice point for which S_a is minimized will have the largest product-of-amplitudes, where for all other lattice points, the product-of-amplitudes will decay to zero in an exponential rate (recall that the normalization operation at the variable node keeps the sum of all amplitudes in a message to be 1). This completes the proof of the second part of the theorem.

We still have to find the vector $\underline{b}$ that minimizes S_a. The basic difference between the case of $d = 2$ and the case of $d > 2$ is that for $d > 2$ we need to analyze the transient behavior of the excitation terms, where for $d = 2$ we only need to analyze the asymptotic behavior, which is much easier to handle.

According to Lemma 9, we have

$$S_a \triangleq \lim_{j\to\infty} \sum_{i=1}^{2n} a_i^{(j)} = \frac{1}{2\sigma^2} (\boldsymbol{G}\underline{b} - \underline{y})^T \boldsymbol{W} (\boldsymbol{G}\underline{b} - \underline{y}) \tag{20}$$

where $\boldsymbol{W}$ is defined by (11) and $\underline{y}$ is the noisy received codeword. Therefore, for $d = 2$, the lattice points whose consistent Gaussians will have largest product-of-amplitudes is the point

for which $(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$ is minimized. This completes the proof of the theorem. □

For $d=2$ we could find an explicit expression for the "winning" lattice point. As discussed above, we could not find an explicit expression for $d>2$, since the result depends on the transient behavior of the excitation sum term, and not only on the steady-state value. However, a reasonable conjecture is to assume that $\underline{b}$ that maximizes the steady-state excitation will also maximize the term that depends on the transient behavior. This means that a reasonable conjecture is to assume that the "winning" lattice point for $d>2$ will also minimize an expression of the form (20).

Note that for $d>2$ we can still show that for "weak" noise, the ML point will have the minimal S. To see that, it comes out from (10) that for zero noise, the ML lattice point will have $a_i^{(t)}=0$ for every t and i, where all other lattice points will have $a_i^{(t)}>0$ for at least some i and t. Therefore, the ML point will have a minimal excitation term along the transient behavior so it will surely have the minimal S and the best product-of-amplitudes. As the noise increases, it is difficult to analyze the transient behavior of $a_i^{(t)}$, as discussed above.

Note that the ML solution minimizes $(\boldsymbol{G}\underline{b}-\underline{y})^T(\boldsymbol{G}\underline{b}-\underline{y})$, where the above analysis yields minimization of $(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$. Obviously, for zero noise (i.e., $\underline{y}=\boldsymbol{G}\cdot\underline{b}$) both minimizations will give the correct solution with zero score. As the noise increases, the solutions may deviate from one another. Therefore, both minimizations will give the same solution for "weak" noise but may give different solutions for "strong" noise.

An example for another decoder that performs this form of minimization is the linear detector, which calculates $\hat{\underline{b}}=\lfloor \boldsymbol{H}\cdot\underline{y}\rceil$ (where $\lfloor x\rceil$ denotes the nearest integer to x). This is equivalent to minimizing $(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$ with $\boldsymbol{W}=\boldsymbol{H}^T\boldsymbol{H}=\boldsymbol{G}^{-1^T}\boldsymbol{G}^{-1}$. The linear detector fails to yield the ML solution if the noise is too strong, due to its inherent noise amplification, and thus fails to attain channel capacity.

For the LDLC iterative decoder, we would like that the deviation from the ML decoder due to the $\boldsymbol{W}$ matrix would be negligible in the expected range of noise variance. Experimental results (see Section IX) show that the iterative decoder indeed converges to the ML solution for noise variance values that approach channel capacity. However, for quantization or shaping applications (see Section VIII-B), where the effective noise is uniformly distributed along the Voronoi cell of the lattice (and is much stronger than the noise variance at channel capacity) the iterative decoder fails, and this can be explained by the influence of the $\boldsymbol{W}$ matrix on the minimization, as described above. Note from (11) that as $\alpha\to 1$, $\boldsymbol{W}$ approaches a scaled identity matrix, which means that the minimization criterion approaches the ML criterion. However, the variances converge as α^t, so as $\alpha\to 1$ convergence time approaches infinity.

Until this point, we concentrated only on consistent Gaussians, and checked what lattice point maximizes the product-of-amplitudes of all the corresponding consistent Gaussians. However, this approach does not necessarily lead to the lattice point that will be finally chosen by the decoder, due to three main reasons.

1) It comes out experimentally that the strongest Gaussian in each message is not necessarily a consistent Gaussian, but a Gaussian that started as nonconsistent and became consistent at a certain iteration. Such a Gaussian will finally converge to the appropriate lattice point, since the convergence of the mean values is independent of initial conditions. The nonconsistency at the first several iterations, where the mean values are still very noisy, allows these Gaussians to accumulate stronger amplitudes than the consistent Gaussians (recall that the exponential weighting in (14) for $d>2$ results in strong dependency on the behavior at the first iterations).
2) There is an exponential number of Gaussians that start as nonconsistent and become consistent (with the same integer vector $\underline{b}$) at a certain iteration, and the final amplitude of the Gaussians at the lattice point coordinates will be determined by the sum of all these Gaussians.
3) We ignored nonconsistent Gaussians that endlessly remain nonconsistent. We have not shown it analytically, but it is reasonable to assume that the excitation terms for such Gaussians will be weaker than for Gaussians that become consistent at some point, so their amplitude will fade away to zero. However, nonconsistent Gaussians are born every iteration, even at steady state. The "newly-born" nonconsistent Gaussians may appear as sidelobes to the main impulse, since it may take several iterations until they are attenuated. Proper choice of the coefficients of $\boldsymbol{H}$ may minimize this effect, as discussed in Sections III-A and V-A. However, these Gaussians may be a problem for small d (e.g., $d=2$) where the product step at the variable node does not include enough messages to suppress them.

Note that the first two issues are not a problem for $d=2$, where the winning lattice point depends only on the asymptotic behavior. The amplitude of a sum of Gaussians that converged to the same coordinates will still be governed by (18) and the winning lattice point will still minimize (20). The third issue is a problem for small d, but less problematic for large d, as described above.

As a result, we cannot regard the convergence analysis of the consistent Gaussians' amplitudes as a complete convergence analysis. However, it can certainly be used as a qualitative analysis that gives certain insights to the convergence process. Two main observations are as follows.

1) The narrow variable node messages tend to converge to single impulses at the coordinates of a single lattice point. This results from (16), (19), which show that the "nonwinning" consistent Gaussians will have amplitudes that decrease to zero relative to the amplitude of the "winning" consistent Gaussian. This result remains valid for the sum of nonconsistent Gaussians that became consistent at a certain point, because it results from the nonstable nature of the recursion (9), which makes strong Gaussians stronger in an exponential manner. The single impulse might be accompanied by weak "sidelobes" due to newly-born nonconsistent Gaussians.

   Interestingly, this form of solution is different from the exact solution (1), where every lattice point is represented

by an impulse at the appropriate coordinate, with amplitude that depends on the Euclidean distance of the lattice point from the observation. The iterative decoder's solution has a single impulse that corresponds to a single lattice point, where all other impulses have amplitudes that decay to zero. This should not be a problem, as long as the ML point is the remaining point (see discussion above).

2) We have shown that for $d = 2$ the strongest consistent Gaussians relate to $\underline{b}$ that minimizes an expression of the form $(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$. We proposed a conjecture that this is also true for $d > 2$. We can further widen the conjecture to say that the finally decoded $\underline{b}$ (and not only the $\underline{b}$ that relates to strongest consistent Gaussians) minimizes such an expression. Such a conjecture can explain why the iterative decoder works well for decoding near channel capacity, but fails for quantization or shaping, where the effective noise variance is much larger.

E. Summary of Convergence Results

The convergence analysis can be summarized as follows. It was first shown that the variable node messages are Gaussian mixtures. Therefore, it is sufficient to analyze the sequences of variances, mean values and relative amplitudes of the Gaussians in each mixture.

Starting with the variances, it was shown that with proper choice of the Latin square LDLC generating sequence, each variable node generates $d-1$ "narrow" messages, whose variance decreases exponentially to zero, and a single "wide" message, whose variance reaches a finite value.

Consistent Gaussians were then defined as Gaussians that their generation process always involved the same integer at the same check node. Consistent Gaussians can then be related to an integer vector $\underline{b}$ or equivalently to the lattice point $\boldsymbol{G}\underline{b}$. It was then shown that under appropriate conditions on $\boldsymbol{H}$, the mean values of consistent Gaussians that belong to narrow messages converge to the coordinates of the appropriate lattice point. The mean values of wide messages also converge to these coordinates, but with a steady-state error.

Then, the amplitudes of consistent Gaussians were analyzed. For row and column degree $d = 2$ it was shown that the consistent Gaussians with maximal product-of-amplitudes (over all messages) are those that correspond to an integer vector $\underline{b}$ than minimizes $(\boldsymbol{G}\underline{b}-\underline{y})^T\boldsymbol{W}(\boldsymbol{G}\underline{b}-\underline{y})$, where $\boldsymbol{W}$ is a positive definite matrix that depends only on $\boldsymbol{H}$. The product-of-amplitudes for all other consistent Gaussians decays to zero.

For $d > 2$ the analysis is complex and depends on the transient behavior of the mean values and variances (and not only on their steady-state values). However, a reasonable conjecture is to assume that a same form of the criterion for $d = 2$ is also minimized for $d > 2$. This criterion is different from the ML lattice point, which minimizes $\left\|\boldsymbol{G}\cdot\underline{b}-\underline{y}\right\|^2$, where both criteria give the same point for weak noise but may give different solutions for strong noise. This may explain the experiments where the iterative decoder is successful in decoding the ML point for the AWGN channel near channel capacity, but fails in quantization or shaping applications where the effective noise is much stronger. The convergence results also indicate that the iterative decoder converges to impulses at the coordinates of a single lattice point.

As explained throughout, analyzing the amplitudes of consistent Gaussians is not sufficient, so these results cannot be regarded as a complete convergence analysis. However, the analysis gave a set of necessary conditions on $\boldsymbol{H}$, and also led to useful insights to the convergence process.

V. Code Design

A. Choosing the Generating Sequence

We shall concentrate on Latin square LDLC, since they have inherent diversity of the nonzero elements in each row and column, which was shown above to be beneficial. It still remains to choose the LDLC generating sequence $h_1, h_2, \ldots, h_d$. Assume that the algorithm converged, and each pdf has a peak at the desired value. When the periodic functions are multiplied at a variable node, the correct peaks will then align. We would like that all the other peaks will be strongly attenuated, i.e., there will be no other point where the peaks align. This resembles the definition of the least common multiple (LCM) of integers: if the periods were integers, we would like to have their LCM as large as possible. This argument suggests the sequence $\{1/2, 1/3, 1/5, 1/7, 1/11, 1/13, 1/17, \ldots\}$, i.e., the reciprocals of the smallest d prime numbers. Since the periods are $1/h_1, 1/h_2, \ldots, 1/h_d$, we will get the desired property. Simulations have shown that increasing d beyond 7 with this choice gave negligible improvement. Also, performance was improved by adding some "dither" to the sequence, resulting in $\{1/2.31, 1/3.17, 1/5.11, 1/7.33, 1/11.71, 1/13.11, 1/17.55\}$. For $d < 7$, the first d elements are used.

An alternative approach is a sequence of the form $\{1, \epsilon, \epsilon, \ldots, \epsilon\}$, where $\epsilon \ll 1$. For this case, every variable node will receive a single message with period 1 and $d-1$ messages with period $1/\epsilon$. For small ϵ, the period of these $d-1$ messages will be large and multiplication by the channel Gaussian will attenuate all the unwanted replicas. The single remaining replica will attenuate all the unwanted replicas of the message with period 1. A convenient choice is $\epsilon = \frac{1}{\sqrt{d}}$, which ensures that $\alpha = \frac{d-1}{d} < 1$, as required by Theorem 1. As an example, for $d = 7$ the sequence will be $\{1, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}\}$.

B. Necessary Conditions on $\boldsymbol{H}$

The Latin square LDLC definition and convergence analysis imply four necessary conditions on $\boldsymbol{H}$.

1) $|\det(\boldsymbol{H})| = 1$. This condition is part of the LDLC definition, which ensures proper density of the lattice points in $\mathbb{R}^m$. If $|\det(\boldsymbol{H})| \neq 1$, it can be easily normalized by dividing $\boldsymbol{H}$ by $\sqrt[n]{|\det(\boldsymbol{H})|}$. Note that practically we can allow $|\det(\boldsymbol{H})| \neq 1$ as long as $\sqrt[n]{|\det(\boldsymbol{H})|} \approx 1$, since $\sqrt[n]{|\det(\boldsymbol{H})|}$ is the gain factor of the transmitted codeword. For example, if $n = 1000$, having $|\det(\boldsymbol{H})| = 0.01$ is acceptable, since we have $\sqrt[n]{|\det(\boldsymbol{H})|} = 0.995$, which means that the codeword has to be further amplified by $20 \cdot \log_{10}(0.995) = 0.04$ dB, which is negligible.

Note that normalizing $\boldsymbol{H}$ is applicable only if $\boldsymbol{H}$ is nonsingular. If $\boldsymbol{H}$ is singular, a row and a column should be sequentially omitted until $\boldsymbol{H}$ becomes full rank. This process may result in slightly reducing n and a slightly different row and column degrees than originally planned.

2) $\alpha < 1$, where $\alpha \triangleq \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$. This guarantees exponential convergence rate for the variances (Theorem 1). Choosing a smaller α results in faster convergence, but we should not take α too small since the steady-state variance of the wide variable node messages, as well as the steady-state error of the mean values of these messages, increases when α decreases, as discussed in Section IV-C. This may result in deviation of the decoded codeword from the ML codeword, as discussed in Section IV-D. For the first LDLC generating sequence of the previous subsection, we have $\alpha = 0.92$ and 0.87 for $d = 7$ and 5, respectively, which is a reasonable trade off. For the second sequence type we have $\alpha = \frac{d-1}{d}$.
3) All the eigenvalues of $\tilde{\boldsymbol{H}}$ must have magnitude less than 1, where $\tilde{\boldsymbol{H}}$ is defined in Theorem 2. This is a necessary condition for convergence of the mean values of the narrow messages. Note that adding random signs to the nonzero $\boldsymbol{H}$ elements is essential to fulfill this necessary condition, as explained in Section IV-C.
4) All the eigenvalues of $\boldsymbol{F}$ must have magnitude less than 1, where $\boldsymbol{F}$ is defined in Theorem 3. This is a necessary condition for convergence of the mean values of the wide messages.

Interestingly, it comes out experimentally that for large codeword length n and relatively small degree d (e.g., $n \geq 1000$ and $d \leq 10$), a Latin square LDLC with generating sequence that satisfies $h_1 = 1$ and $\alpha < 1$ results in $\boldsymbol{H}$ that satisfies all these four conditions: $\boldsymbol{H}$ is nonsingular without any need to omit rows and columns, $\sqrt[n]{|\det(\boldsymbol{H})|} \approx 1$ without any need for normalization, and all eigenvalues of $\tilde{\boldsymbol{H}}$ and $\boldsymbol{F}$ have magnitude less than 1 (typically, the largest eigenvalue of $\tilde{\boldsymbol{H}}$ or $\boldsymbol{F}$ has magnitude from 0.94 to 0.97, almost independently of n and the choice of nonzero $\boldsymbol{H}$ locations). Therefore, by simply dividing the first generating sequence of the previous subsection by its first element, the constructed $\boldsymbol{H}$ meets all the necessary conditions, where the second type of sequence meets the conditions without any need for modifications.

C. Construction of $\boldsymbol{H}$ for Latin Square LDLC

We shall now present a simple algorithm for constructing a parity check matrix for a Latin square LDLC. If we look at the bipartite graph, each variable node and each check node has d edges connected to it, one with every possible weight $h_1, h_2, \ldots, h_d$. All the edges that have the same weight h_j form a permutation from the variable nodes to the check nodes (or vice versa). The proposed algorithm generates d random permutations and then searches sequentially and cyclically for 2-loops (two parallel edges from a variable node to a check node) and 4-loops (two variable nodes that both are connected to a pair of check nodes). When such a loop is found, a pair is swapped in one of the permutations such that the loop is removed. A detailed pseudo-code for this algorithm is given in Appendix VII.

VI. Decoder Implementation

For a practical implementation, each pdf should be approximated by a discrete vector with resolution Δ and finite range. It is natural to choose a finite range which is symmetric around the noisy channel observation. This truncation may generate errors if the actual codeword coordinate is located outside this finite range, i.e., the Gaussian channel noise sample for a specific coordinate was larger than half the finite range. According to the Gaussian Q-function, choosing a range of, say, 6σ to both sides of the noisy channel observation will ensure that the error probability due to pdf truncation will be $\approx 10^{-9}$. Near capacity, $\sigma^2 \approx \frac{1}{2\pi e}$, so $12\sigma \approx 3$. Simulation showed that resolution errors became negligible for $\Delta = 1/64$. Each pdf was then stored in a $L = 256$ elements vector, corresponding to a range of size 4.

At the check node, the pdf $f_j(x)$ that arrives from variable node j is first expanded by h_j (the appropriate coefficient of $\boldsymbol{H}$) to get $f_j(x/h_j)$. In a discrete implementation with resolution Δ the pdf is a vector of values $f_j(k\Delta)$, $k \in \mathbb{Z}$. As described in Section V, we shall usually use $h_j \leq 1$ so the expanded pdf will be shorter than the original pdf. If the expand factor $1/|h_j|$ was an integer, we could simply decimate $f_j(k\Delta)$ by $1/|h_j|$. However, in general it is not an integer so we should use some kind of interpolation. The pdf $f_j(x)$ is certainly not band limited, and as the iterations go on it approaches an impulse, so simple interpolation methods (e.g., linear) are not suitable. Suppose that we need to calculate $f_j((k+\epsilon)\Delta)$, where $-0.5 \leq \epsilon \leq 0.5$. A simple interpolation method which showed to be effective is to average $f_j(x)$ around the desired point, where the averaging window length l_w is chosen to ensure that every sample of $f_j(x)$ is used in the interpolation of at least one output point. This ensures that an impulse cannot be missed. The interpolation result is then $\frac{1}{2l_w+1} \sum_{i=-l_w}^{l_w} f_j((k-i)\Delta)$, where $l_w = \left\lfloor \frac{\lceil 1/|h_j| \rceil}{2} \right\rfloor$.

The most computationally extensive step at the check nodes is the calculation the convolution of $d-1$ expanded pdfs. An efficient method is to calculate the fast Fourier transforms (FFTs) of all the pdfs, multiply the results and then perform inverse FFT (IFFT). The resolution of the FFT should be larger than the expected convolution length, which is roughly $L_{\text{out}} \approx L \cdot \sum_{i=1}^{d} h_i$, where L denotes the original pdf length. Appendix VIII shows a way to use FFTs of size $1/\Delta$, where Δ is the resolution of the pdf. Usually $1/\Delta \ll L_{\text{out}}$ so FFT complexity is significantly reduced. Practical values are $L = 256$ and $\Delta = 1/64$, which give an improvement factor of at least 4 in complexity.

Each variable node receives d check node messages. The output variable node message is calculated by generating the product of $d-1$ input messages and the channel Gaussian. As the iterations go on, the messages get narrow and may become impulses, with only a single nonzero sample. Quantization effects may cause impulses in two messages to be shifted by one sample. This will result in a zero output (instead of an impulse). Therefore, it was found useful to widen each check node message $Q(k)$ prior to multiplication, such that $Q_w(k) = \sum_{i=-1}^{1} Q(k+i)$, i.e., the message is added to its right shifted and left shifted (by one sample) versions.

VII. Computational Complexity and Storage Requirements

Most of the computational effort is invested in the d FFTs and d IFFTs (of length $1/\Delta$) that each check node performs each iteration. The total number of multiplications for t iterations is $O\left(n \cdot d \cdot t \cdot \frac{1}{\Delta} \cdot \log_2(\frac{1}{\Delta})\right)$. As in binary LDPC codes, the computational complexity has the attractive property of being linear with block length. However, the constant that precedes the linear term is significantly higher, mainly due to the FFT operations.

The memory requirements are governed by the storage of the nd check node and variable node messages, with total memory of $O(n \cdot d \cdot L)$. Compared to binary LDPC, the factor of L significantly increases the required memory. For example, for $n = 10\,000$, $d = 7$ and $L = 256$, the number of storage elements is of the order of 10^7.

VIII. Encoding and Shaping

A. Encoding

The LDLC encoder has to calculate $\underline{x} = \boldsymbol{G} \cdot \underline{b}$, where $\underline{b}$ is an integer message vector. Note that unlike $\boldsymbol{H}$, $\boldsymbol{G} = \boldsymbol{H}^{-1}$ is not sparse, in general, so the calculation requires computational complexity and storage of $O(n^2)$. This is not a desirable property because the decoder's computational complexity is only $O(n)$. A possible solution is to use the Jacobi method [26] to solve $\boldsymbol{H} \cdot \underline{x} = \underline{b}$, which is a system of sparse linear equations. Using this method, a Latin square LDLC encoder calculates several iterations of the form

$$\underline{x}^{(t)} = \tilde{\underline{b}} - \tilde{\boldsymbol{H}} \cdot \underline{x}^{(t-1)} \tag{21}$$

with initialization $\underline{x}^{(0)} = \underline{0}$. The matrix $\tilde{\boldsymbol{H}}$ is defined in Lemma 6 of Section IV-C. The vector $\tilde{\underline{b}}$ is a permuted and scaled version of the integer vector $\underline{b}$, such that the ith element of $\tilde{\underline{b}}$ equals the element of $\underline{b}$ for which the appropriate row of $\boldsymbol{H}$ has its largest magnitude value at the ith location. This element is further divided by this largest magnitude element.

A necessary and sufficient condition for convergence to $\underline{x} = \boldsymbol{G} \cdot \underline{b}$ is that all the eigenvalues of $\tilde{\boldsymbol{H}}$ have magnitude less than 1 [26]. However, it was shown that this is also a necessary condition for convergence of the LDLC iterative decoder (see Sections IV-C, V-B), so it is guaranteed to be fulfilled for a properly designed Latin square LDLC. Since $\tilde{\boldsymbol{H}}$ is sparse, this is an $O(n)$ algorithm, both in complexity and storage.

B. Shaping

For practical use with the power constrained AWGN channel, the encoding operation must be accompanied by shaping, in order to prevent the transmitted codeword's power from being too large. Therefore, instead of mapping the information vector $\underline{b}$ to the lattice point $\underline{x} = \boldsymbol{G} \cdot \underline{b}$, it should be mapped to some other lattice point $\underline{x}' = \boldsymbol{G} \cdot \underline{b}'$, such that the lattice points that are used as codewords belong to a shaping region (e.g., an n-dimensional sphere). The shaping operation is the mapping of the integer vector $\underline{b}$ to the integer vector $\underline{b}'$.

As explained in Section II-A, this work concentrates on the lattice design and the lattice decoding algorithm, and not on the shaping region or shaping algorithms. Therefore, this section will only highlight some basic shaping principles and ideas.

A natural shaping scheme for lattice codes is nested lattice coding [12]. In this scheme, shaping is done by quantizing the lattice point $\boldsymbol{G} \cdot \underline{b}$ onto a coarse lattice $\boldsymbol{G}'$, where the transmitted codeword is the quantization error, which is uniformly distributed along the Voronoi cell of the coarse lattice. If the second moment of this Voronoi cell is close to that of an n-dimensional sphere, the scheme will attain close-to-optimal shaping gain. Specifically, assume that the information vector $\underline{b}$ assumes integer values in the range $0, 1, \ldots, M-1$ for some constant integer M. Then, we can choose the coarse lattice to be $\boldsymbol{G}' = M\boldsymbol{G}$. The volume of the Voronoi cell for this lattice is M^n, since we assume $\det(\boldsymbol{G}) = 1$ (see Section II-A). If the shape of the Voronoi cell resembles an n-dimensional sphere (as expected from a capacity approaching lattice code), it will attain optimal shaping gain (compared to uncoded transmission of the original integer sequence $\underline{b}$).

The shaping operation will find the coarse lattice point $M\boldsymbol{G}\underline{k}$, $\underline{k} \in \mathbb{Z}^n$, which is closest to the fine lattice point $\underline{x} = \boldsymbol{G} \cdot \underline{b}$. The transmitted codeword will be

$$\underline{x}' = \underline{x} - M\boldsymbol{G}\underline{k} = \boldsymbol{G}(\underline{b} - M\underline{k}) = \boldsymbol{G}\underline{b}'$$

where $\underline{b}' \triangleq \underline{b} - M\underline{k}$ (note that the "inverse shaping" at the decoder, i.e., transforming from $\underline{b}'$ to $\underline{b}$, is a simple modulo calculation: $\underline{b} = \underline{b}' \bmod M$). Finding the closest coarse lattice point $M\boldsymbol{G}\underline{k}$ to $\underline{x}$ is equivalent to finding the closest fine lattice point $\boldsymbol{G}\cdot\underline{k}$ to the vector $\underline{x}/M$. This is exactly the operation of the iterative LDLC decoder, so we could expect that is could be used for shaping. However, simulations show that the iterative decoding finds a vector $\underline{k}$ with poor shaping gain. The reason is that for shaping, the effective noise is much stronger than for decoding, and the iterative decoder fails to find the nearest lattice point if the noise is too large (see Section IV-D).

Therefore, an alternative algorithm has to be used for finding the nearest coarse lattice point. The complexity of finding the nearest lattice point grows exponentially with the lattice dimension n and is not feasible for large dimensions [27]. However, unlike decoding, for shaping applications it is not critical to find the exact nearest lattice point, and approximate algorithms may be considered (see [16]). A possible method [28] is to perform QR decomposition on $\boldsymbol{G}$ in order to transform to a lattice with upper triangular generator matrix, and then use sequential decoding algorithms (such as the Fano algorithm) to search the resulting tree. The main disadvantage of this approach is computational complexity and storage of at least $O(n^2)$. Finding an efficient shaping scheme for LDLC is certainly a topic for further research.

IX. Simulation Results

Latin square LDLC with the first generating sequence of Section V-A (i.e., $\{1/2.31, 1/3.17, 1/5.11, 1/7.33, 1/11.71, 1/13.11, 1/17.55\}$) were simulated for the AWGN channel at various block lengths. The degree was $d = 5$ for $n = 100$ and $d = 7$ for all other n. For $n = 100$ the matrix $\boldsymbol{H}$ was further normalized to get $\sqrt[n]{\det(\boldsymbol{H})} = 1$. For all other n, normalizing the generating sequence such that the largest

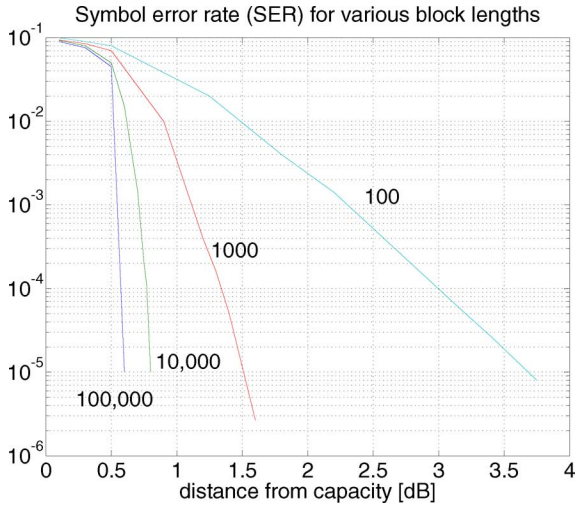

Fig. 4. Simulation results.

element has magnitude 1 also gave the desired determinant normalization (see Section V-B). The $\boldsymbol{H}$ matrices were generated using the algorithm of Section V-C. The probability density function (pdf) resolution was set to $\Delta = 1/256$ with a total range of 4, i.e., each pdf was represented by a vector of $L = 1024$ elements. High resolution was used since our main target is to prove the LDLC concept and eliminate degradation due to implementation considerations. For this reason, the decoder was used with 200 iterations (though most of the time, a much smaller number was sufficient).

In all simulations the all-zero codeword was used. Approaching channel capacity is equivalent to $\sigma^2 \to \frac{1}{2\pi e}$ (see Section II-A), so performance is measured in symbol error rate (SER), versus the distance of the noise variance σ^2 from capacity (in dB). The results are shown in Fig. 4. At SER of 10^{-5}, for $n = 100\,000, 10\,000, 1000, 100$ we can work as close as 0.6, 0.8, 1.5, and 3.7 dB from capacity, respectively.

Similar results were obtained for $d = 7$ with the second type of generating sequence of Section V-A, i.e., $\{1, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}\}$. Results were slightly worse than for the first generating sequence (by less than 0.1 dB). Increasing d did not give any visible improvement.

We shall now compare these results with the LDPC-based coding schemes that were described in Section I: LDPC-based multi-level coding [19], [29], nonbinary alphabet LDPC coding [18] and LDPC lattice coding [15]. The comparison is summarized in Table I. The table shows the simulation results which are reported for each scheme in the appropriate reference, including codeword length n, distance from channel capacity and error performance, measured as either bit error rate (BER), symbol error rate (SER) or normalized error probability (NEP), as defined in [15]. Note that the schemes of [29] and [18] already include an inherent shaping operation, so the simulation results for these schemes already include practical shaping performance.

It can be seen that compared to the other schemes, LDLC have very good performance, and they offer an attractive tradeoff between performance and codeword length. Regarding complexity, all the schemes have linear complexity. As described above, the linear coefficient tends to be higher for LDLC, due to the FFT operations. However, as the number of information bits per channel use increases, LDLC complexity becomes comparable to the other schemes. Furthermore, we currently investigate techniques that will reduce the computational complexity of LDLC decoding, as well as design methodologies for the parity check matrix, so we hope that LDLC performance and complexity will further improve.

TABLE I
COMPARISON OF CODING SCHEMES

coding scheme	n [symbols]	error probability	distance from capacity [dB]
LDPC based multilevel coding [19]	10,000	BER = 10^{-5}	1.0
LDPC based multilevel coding [29]	320,000	BER = 10^{-5}	0.6
non-binary alphabet LDPC coding [18]	180,000	SER = 10^{-6}	0.56
LDPC lattice [15]	2,000	NEP = 10^{-6}	3.0
LDLC	100,000	SER = 10^{-5}	0.6

X. CONCLUSION

LDLC are novel lattice codes that can approach capacity and be decoded efficiently. Good error performance within ~ 0.5 dB from capacity at block length of 100 000 symbols was demonstrated. Convergence analysis was presented for the iterative decoder, which is not complete, but yields necessary conditions on $\boldsymbol{H}$ and significant insight to the convergence process. Code parameters were chosen from intuitive arguments, so it is reasonable to assume that when the code structure will be more understood, better parameters could be found, and channel capacity could be approached even closer.

Multiple-input–multiple-output (MIMO) communication systems have become popular in recent years. Lattice codes have been proposed in this context as space–time codes (LAST) [30]. The concatenation of the lattice encoder and the MIMO channel generates a lattice. If LDLC are used as lattice codes and the MIMO configuration is small, the inverse generator matrix of this concatenated lattice can be assumed to be sparse. Therefore, the MIMO channel and the LDLC can be jointly decoded using an LDLC-like decoder. However, even if a Latin square LDLC is used as the lattice code, the concatenated lattice is not guaranteed to be equivalent to a Latin square LDLC, and the necessary conditions for convergence are not guaranteed to be fulfilled. Therefore, the usage of LDLC for MIMO systems is a topic for further research.

APPENDIX I
EXACT PDF CALCULATIONS

Given the n-dimensional noisy observation $\underline{y} = \underline{x} + \underline{w}$ of the transmitted codeword $\underline{x} = \boldsymbol{G}\underline{b}$, we would like to calculate the probability density function (pdf) $f_{x_k|\underline{y}}(x_k|\underline{y})$. We shall start by calculating $f_{\underline{x}|\underline{y}}(\underline{x}|\underline{y}) = \frac{f_{\underline{x}}(\underline{x}) f_{\underline{y}|\underline{x}}(\underline{y}|\underline{x})}{f_{\underline{y}}(\underline{y})}$. Denote the shaping region by B ($\boldsymbol{G}$ will be used to denote both the lattice and its

generator matrix). $f_{\underline{x}}(\underline{x})$ is a sum of $|\boldsymbol{G} \cap B|$ n-dimensional Dirac delta functions, since $\underline{x}$ has nonzero probability only for the lattice points that lie inside the shaping region. Assuming further that all codewords are used with equal probability, all these delta functions have equal weight of $\frac{1}{|\boldsymbol{G} \cap B|}$. The expression for $f_{\underline{y}|\underline{x}}(\underline{y}|\underline{x})$ is simply the pdf of the i.i.d. Gaussian noise vector. We therefore get

$$\begin{aligned} & f_{\underline{x}|\underline{y}}(\underline{x}|\underline{y}) \\ &= \frac{f_{\underline{x}}(\underline{x}) f_{\underline{y}|\underline{x}}(\underline{y}|\underline{x})}{f_{\underline{y}}(\underline{y})} \\ &= \frac{\frac{1}{|\boldsymbol{G} \cap B|} \sum_{\underline{l} \in \boldsymbol{G} \cap B} \delta(\underline{x} - \underline{l}) \cdot (2\pi\sigma^2)^{-n/2} e^{-\sum_{i=1}^{n} (y_i - x_i)^2 / 2\sigma^2}}{f_{\underline{y}}(\underline{y})} \\ &= C \cdot \sum_{\underline{l} \in \boldsymbol{G} \cap B} \delta(\underline{x} - \underline{l}) \cdot e^{-d^2(\underline{l}, \underline{y})/2\sigma^2} \end{aligned} \tag{22}$$

where C is not a function of $\underline{x}$ and $d^2(\underline{l}, \underline{y})$ is the squared Euclidean distance between the vectors $\underline{l}$ and $\underline{y}$ in $\mathbb{R}^n$. It can be seen that the conditional pdf of $\underline{x}$ has a delta function for each lattice point, located at this lattice point with weight that is proportional to the exponent of the negated squared Euclidean distance of this lattice point from the noisy observation. The ML point corresponds to the delta function with largest weight.

As the next step, instead of calculating the n-dimensional pdf of the whole vector $\underline{x}$, we shall calculate the n one-dimensional marginal pdfs for each of the components x_k of the vector $\underline{x}$ (conditioned on the whole observation vector $\underline{y}$)

$$\begin{aligned} & f_{x_k|\underline{y}}(x_k|\underline{y}) \\ &= \int\int_{x_i, i \neq k} \cdots \int f_{\underline{x}|\underline{y}}(\underline{x}|\underline{y}) dx_1 dx_2 \cdots dx_{k-1} dx_{k+1} \cdots dx_n \\ &= C \cdot \sum_{\underline{l} \in \boldsymbol{G} \cap B} \delta(x_k - l_k) \cdot e^{-d^2(\underline{l}, \underline{y})/2\sigma^2}. \end{aligned} \tag{23}$$

This finishes the proof of (1). It can be seen that the conditional pdf of x_k has a delta function for each lattice point, located at the projection of this lattice point on the coordinate x_k, with weight that is proportional to the exponent of the negated squared Euclidean distance of this lattice point from the noisy observation. The ML point will therefore correspond to the delta function with largest weight in each coordinate. Note, however, that if several lattice points have the same projection on a specific coordinate, the weights of the corresponding delta functions will add and may exceed the weight of the ML point.

Appendix II
Extending Gallager's Technique to the Continuous Case

In [5], the derivation of the LDPC iterative decoder was simplified using the following technique: the codeword elements x_k were assumed i.i.d. and a condition was added to all the probability calculations, such that only valid codewords were actually considered. The question is then how to choose the marginal pdf of the codeword elements. In [5], binary codewords were considered, and the i.i.d distribution assumed the values "0" and "1" with equal probability. Since we extend the technique to the continuous case, we have to set the continuous marginal distribution $f_{x_k}(x_k)$. It should be set such that $f_{\underline{x}}(\underline{x})$, assuming that $\underline{x}$ is a lattice point, is the same as $f(\underline{x}|\underline{s} \in \mathbb{Z}^n)$, assuming that x_k are i.i.d with marginal pdf $f_{x_k}(x_k)$, where $\underline{s} \triangleq \boldsymbol{H} \cdot \underline{x}$. This $f_{\underline{x}}(\underline{x})$ equals a weighted sum of Dirac delta functions at all lattice points, where the weight at each lattice point equals the probability to use this point as a codeword.

Before proceeding, we need the following property of conditional probabilities. For any two continuous valued RV's u, v we have

$$f(u|v \in \{v_1, v_2, \ldots, v_N\}) = \frac{\sum_{k=1}^{N} f_{u,v}(u, v_k)}{\sum_{k=1}^{N} f_v(v_k)}. \tag{24}$$

(This property can be easily proved by following the lines of [25, pp. 159-160], and can also be extended to the infinite sum case).

Using (24), we now have

$$\begin{aligned} f(\underline{x}|\underline{s} \in \mathbb{Z}^n) &= \frac{\sum_{\underline{i} \in \mathbb{Z}^n} f_{\underline{x},\underline{s}}(\underline{x}, \underline{s} = \underline{i})}{\sum_{\underline{i} \in \mathbb{Z}^n} f_{\underline{s}}(\underline{i})} \\ &= C \sum_{\underline{i} \in \mathbb{Z}^n} f(\underline{x}) f(\underline{s} = \underline{i}|\underline{x}) \\ &= C' \sum_{\underline{i} \in \mathbb{Z}^n} f(\underline{x}) \delta(\underline{x} - \boldsymbol{G}\underline{i}) \end{aligned} \tag{25}$$

where C, C' are independent of $\underline{x}$.

The result is a weighted sum of Dirac delta functions at all lattice points, as desired. Now, the weight at each lattice point should equal the probability to use this point as a codeword. Therefore, $f_{x_k}(x_k)$ should be chosen such that at each lattice point, the resulting vector distribution $f_{\underline{x}}(\underline{x}) = \prod_{k=1}^{n} f_{x_k}(x_k)$ will have a value that is proportional to the probability to use this lattice point. At $\underline{x}$ which is not a lattice point, the value of $f_{\underline{x}}(\underline{x})$ is not important.

Appendix III
Derivation of the Iterative Decoder

In this appendix we shall derive the LDLC iterative decoder for a code with dimension n, using the tree assumption and Gallager's trick.

Referring to Fig. 2, assume that there are only 2 tiers. Using Gallager's trick we assume that the x_k's are i.i.d. We would like to calculate $f(x_1|\underline{y}, \underline{s} \in \mathbb{Z}^n)$, where $\underline{s} \triangleq \boldsymbol{H} \cdot \underline{x}$. Due to the tree assumption, we can do it in two steps:

1) calculate the conditional pdf of the tier 1 variables of x_1, conditioned only on the check equations that relate the tier 1 and tier 2 variables;
2) calculate the conditional pdf of x_1 itself, conditioned only on the check equations that relate x_1 and its first tier variables, but using the results of step 1 as the pdfs for the tier 1 variables. Hence, the results will be equivalent to conditioning on all the check equations.

There is a basic difference between the calculation in step 1 and step 2: the condition in step 2 involves all the check equations that are related to x_1, where in step 1 a single check equation is always omitted (the one that relates the relevant tier 1 element with x_1 itself).

Assume now that there are many tiers, where each tier contains distinct elements of $\underline{x}$ (i.e., each element appears only once in the resulting tree). We can then start at the farthest tier and start moving toward x_1. We do it by repeatedly calculating step 1. After reaching tier 1, we use step 2 to finally calculate the desired conditional pdf for x_1.

This approach suggests an iterative algorithm for the calculation of $f(x_k|(\underline{y}, \underline{s} \in \mathbb{Z}^n)$ for $k = 1, 2, \ldots, n$. In this approach we assume that the resulting tier diagram for each x_k contains distinct elements for several tiers (larger or equal to the number of required iterations). We then repeat step 1 several times, where the results of the previous iteration are used as initial pdfs for the next iteration. Finally, we perform step 2 to calculate the final results.

Note that by conditioning only on part of the check equations in each iteration, we cannot restrict the result to the shaping region. This is the reason that the decoder performs lattice decoding and not exact ML decoding, as described in Section III.

We shall now turn to derive the basic iteration of the algorithm. For simplicity, we shall start with the final step of the algorithm (denoted step 2 above). We would like to perform t iterations, so assume that for each x_k there are t tiers with a total of N_c check equations. For every x_k we need to calculate

$$f(x_k|\underline{s}\in\mathbb{Z}^{N_c}, \underline{y}) = f(x_k|\underline{s}^{(\text{tier}_1)}\in\mathbb{Z}^{c_k}, \underline{s}^{(\text{tier}_2:\text{tier}_t)}\in\mathbb{Z}^{N_c-c_k}, \underline{y})$$

where c_k is the number of check equations that involve x_k. $\underline{s}^{(\text{tier}_1)} \triangleq \boldsymbol{H}^{(\text{tier}_1)} \cdot \underline{x}$ denotes the value of the left hand side of these check equations when $\underline{x}$ is substituted ($\boldsymbol{H}^{(\text{tier}_1)}$ is a submatrix of $\boldsymbol{H}$ that contains only the rows that relate to these check equations), and $\underline{s}^{(\text{tier}_2:\text{tier}_t)}$ relates in the same manner to all the other check equations. For simplicity of notations, denote the event $\underline{s}^{(\text{tier}_2:\text{tier}_t)} \in \mathbb{Z}^{N_c-c_k}$ by A. As explained above, in all the calculations we assume that all the x_k's are independent.

Using (24), we get

$$f(x_k|\underline{s}^{(\text{tier}_1)} \in \mathbb{Z}^{c_k}, A, \underline{y}) = \frac{\sum_{\underline{i}\in\mathbb{Z}^{c_k}} f(x_k, \underline{s}^{(\text{tier}_1)} = \underline{i}|A, \underline{y})}{\sum_{\underline{i}\in\mathbb{Z}^{c_k}} f(\underline{s}^{(\text{tier}_1)} = \underline{i}|A, \underline{y})}. \tag{26}$$

Evaluating the term inside the sum of the numerator, we get

$$f(x_k, \underline{s}^{(\text{tier}_1)} = \underline{i}|A, \underline{y}) = f(x_k|A, \underline{y}) \cdot f(\underline{s}^{(\text{tier}_1)} = \underline{i}|x_k, A, \underline{y}). \tag{27}$$

Evaluating the left term of (27), we get

$$\begin{aligned} f(x_k|A, \underline{y}) = f(x_k|y_k) &= \frac{f(x_k)f(y_k|x_k)}{f(y_k)} \\ &= \frac{f(x_k)}{f(y_k)} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k-x_k)^2}{2\sigma^2}} \end{aligned} \tag{28}$$

where $f(x_k|\underline{y}) = f(x_k|y_k)$ due to the i.i.d. assumption. Evaluating now the right term of (27), we get

$$f(\underline{s}^{(\text{tier}_1)} = \underline{i}|x_k, A, \underline{y}) = \prod_{m=1}^{c_k} f(s_m^{(\text{tier}_1)} = i_m|x_k, A, \underline{y}) \tag{29}$$

where $s_m^{(\text{tier}_1)}$ denotes the mth component of $\underline{s}^{(\text{tier}_1)}$ and i_m denotes the mth component of $\underline{i}$. Note that each element of $\underline{s}^{(\text{tier}_1)}$ is a linear combination of several elements of $\underline{x}$. Due to the tree assumption, two such linear combinations have no common elements, except for x_k itself, which appears in all linear combinations. However, x_k is given, so the i.i.d assumption implies that all these linear combinations are independent, so (29) is justified. The condition A (i.e., $\underline{s}^{(\text{tier}_2:\text{tier}_t)} \in \mathbb{Z}^{N_c-c_k}$) does not impact the independence due to the tree assumption.

Substituting (27), (28), (29) back in (26), we get

$$\begin{aligned} &f(x_k|\underline{s}^{(\text{tier}_1)}\in\mathbb{Z}^{c_k}, A, \underline{y}) \\ &= C\cdot f(x_k)\cdot e^{-\frac{(y_k-x_k)^2}{2\sigma^2}} \sum_{\underline{i}\in\mathbb{Z}^{c_k}} \prod_{m=1}^{c_k} f(s_m^{(\text{tier}_1)}=i_m|x_k, A, \underline{y}) \\ &= C\cdot f(x_k)\cdot e^{-\frac{(y_k-x_k)^2}{2\sigma^2}} \sum_{i_1\in\mathbb{Z}} \sum_{i_2\in\mathbb{Z}} \cdots \\ &\quad \cdots \sum_{i_{c_k}\in\mathbb{Z}} \prod_{m=1}^{c_k} f(s_m^{(\text{tier}_1)} = i_m|x_k, A, \underline{y}) \\ &= C\cdot f(x_k)\cdot e^{-\frac{(y_k-x_k)^2}{2\sigma^2}} \prod_{m=1}^{c_k} \sum_{i_m\in\mathbb{Z}} f(s_m^{(\text{tier}_1)}=i_m|x_k, A, \underline{y}) \end{aligned} \tag{30}$$

where C is independent of x_k.

We shall now examine the term inside the sum: $f(s_m^{(\text{tier}_1)} = i_m|x_k, A, \underline{y})$. Denote the linear combination that $s_m^{(\text{tier}_1)}$ represents by

$$s_m^{(\text{tier}_1)} = h_{m,1}x_k + \sum_{l=2}^{r_m} h_{m,l}x_{j_l} \tag{31}$$

where $\{h_{m,l}\}$, $l = 1, 2, \ldots, r_m$ is the set of nonzero coefficients of the appropriate parity check equation, and j_l is the set of indices of the appropriate $\underline{x}$ elements (note that the set j_l depends on m but we omit the "m" index for clarity of notations). Without loss of generality, $h_{m,1}$ is assumed to be the coefficient of x_k. Define $z_m \triangleq \sum_{l=2}^{r_m} h_{m,l}x_{j_l}$, such that $s_m^{(\text{tier}_1)} = h_{m,1}x_k + z_m$. We then have

$$\begin{aligned} f(s_m^{(\text{tier}_1)}&=i_m|x_k, A, \underline{y}) \\ &= f_{z_m|x_k,A,\underline{y}}(z_m = i_m - h_{m,1}x_k|x_k, A, \underline{y}). \end{aligned} \tag{32}$$

Now, since we assume that the elements of $\underline{x}$ are independent, the pdf of the linear combination z_m equals the convolution of the pdfs of its components

$$\begin{aligned} &f_{z_m|x_k,A,\underline{y}}(z_m|x_k, A, \underline{y}) \\ &\quad = \frac{1}{|h_{m,2}|} f_{x_{j_2}|A,\underline{y}}\left(\frac{z_m}{h_{m,2}}|A, \underline{y}\right) \\ &\qquad * \frac{1}{|h_{m,3}|} f_{x_{j_3}|A,\underline{y}}\left(\frac{z_m}{h_{m,3}}|A, \underline{y}\right) \\ &\qquad \cdots * \frac{1}{|h_{m,r_m}|} f_{x_{j_{r_m}}|A,\underline{y}}\left(\frac{z_m}{h_{m,r_m}}|A, \underline{y}\right). \end{aligned} \tag{33}$$

Note that the functions $f_{x_{j_i}|\underline{y}}(x_{j_i}|A, \underline{y})$ are simply the output pdfs of the previous iteration.

Define now

$$p_m(x_k) \triangleq f_{z_m|x_k,A,\underline{y}}(z_m = -h_{m,1}x_k|x_k, A, \underline{y}). \tag{34}$$

Substituting (32), (34) in (30), we finally get:

$$f(x_k|\underline{s}^{(\text{tier}_1)} \in \mathbb{Z}^{c_k}, A, \underline{y}) = C \cdot f(x_k) \cdot e^{-\frac{(y_k - x_k)^2}{2\sigma^2}} \prod_{m=1}^{c_k} \sum_{i_m \in \mathbb{Z}} p_m(x_k - \frac{i_m}{h_{m,1}}). \quad (35)$$

This result can be summarized as follows. For each of the c_k check equations that involve x_k, the pdfs (previous iteration results) of the active equation elements, except for x_k itself, are expanded and convolved, according to (33). The convolution result is scaled by $(-h_{m,1})$, the negated coefficient of x_k in this check equation, according to (34), to yield $p_m(x_k)$. Then, a periodic function with period $1/|h_{m,1}|$ is generated by adding an infinite number of shifted versions of the scaled convolution result, according to the sum term in (35). After repeating this process for all the c_k check equations that involve x_k, we get c_k periodic functions, with possibly different periods. We then multiply all these functions. The multiplication result is further multiplied by the channel Gaussian pdf term $e^{-\frac{(y_k - x_k)^2}{2\sigma^2}}$ and finally by $f(x_k)$, the marginal pdf of x_k under the i.i.d assumption. As discussed in Section III, we assume that $f(x_k)$ is a uniform distribution with large enough range. This means that $f(x_k)$ is constant over the valid range of x_k, and can therefore be omitted from (35) and absorbed in the constant C.

As noted above, this result is for the final step (equivalent to step 2 above), where we determine the pdf of x_k according to the pdfs of all its tier 1 elements. However, the repeated iteration step is equivalent to step 1 above. In this step, we assume that x_k is a tier 1 element of another element, say x_l, and derive the pdf of x_k that should be used as input to step 2 of x_l (see Fig. 2). It can be seen that the only difference between step 2 and step 1 is that in step 2 all the check equations that involve x_k are used, where in step 1 the check equation that involves both x_k and x_l is ignored (there must be such an equation since x_k is one of the tier 1 elements of x_l). Therefore, the step 1 iteration is identical to (35), except that the product does not contain the term that corresponds to the check equation that combines both x_k and x_l. Denote

$$f_{kl}(x_k) \triangleq f(x_k|\underline{s}^{(\text{tier}_1 \text{ except } l)} \in \mathbb{Z}^{c_k - 1}, A, \underline{y}). \quad (36)$$

We then get

$$f_{kl}(x_k) = C \cdot e^{-\frac{(y_k - x_k)^2}{2\sigma^2}} \prod_{\substack{m=1 \\ m \neq m_l}}^{c_k} \sum_{i_m \in \mathbb{Z}} p_m(x_k - \frac{i_m}{h_{m,1}}) \quad (37)$$

where m_l is the index of the check equation that combines both x_k and x_l. In principle, a different $f_{kl}(x_k)$ should be calculated for each x_l for which x_k is a tier 1 element. However, the calculation is the same for all x_l that share the same check equation. Therefore, we should calculate $f_{kl}(x_k)$ once for each check equation that involves x_k. l can be regarded as the index of the check equation within the set of check equations that involve x_k.

We can now formulate the iterative decoder. The decoder state variables are pdfs of the form $f_{kl}^{(t)}(x_k)$, where $k = 1, 2, \ldots, n$. For each k, l assumes the values $1, 2, \ldots, c_k$, where c_k is the number of check equations that involve x_k. t denotes the iteration index. For a regular LDLC with degree d there will be nd pdfs. The pdfs are initialized by assuming that x_k is a leaf of the tier diagram. Such a leaf has no tier 1 elements, so $f_{kl}(x_k) = f(x_k) \cdot f(y_k|x_k)$. As explained above for (35), we shall omit the term $f(x_k)$, resulting in initialization with the channel noise Gaussian around the noisy observation y_k. Then, the pdfs are updated in each iteration according to (37). The variable node messages should be further normalized in order to get actual pdfs, such that $\int_{-\infty}^{\infty} f_{kl}(x_k) dx_k = 1$ (this will compensate for the constant C). The final pdfs for x_k, $k = 1, 2, \ldots, n$ are then calculated according to (35).

In Section IV, it is shown that under certain conditions, each variable node receives a single check node message which approaches an impulse, and several other messages that approach a Gaussian with nonzero variance. The source of the impulse message for each variable node is known and depends only on the relevant coefficient of $\boldsymbol{H}$. If enough iterations are performed, this observation suggests an alternative to (35) for the final pdf calculation: instead of multiplying all the incoming messages altogether, the decoder can simply choose the message that approaches an impulse.

Finally, we have to estimate the integer valued information vector $\underline{b}$. This can be done by first estimating the codeword vector $\underline{x}$ from the peaks of the pdfs: $\hat{x_k} = arg\max_{x_k} f(x_k|\underline{s}^{(\text{tier}_1)} \in \mathbb{Z}^{c_k}, A, \underline{y})$. Finally, we can estimate $\underline{b}$ as $\hat{\underline{b}} = \lfloor \boldsymbol{H} \hat{\underline{x}} \rceil$.

An alternative way to recover the integer valued information vector $\underline{b}$ is to estimate the pdfs of the elements of the vector $\boldsymbol{H}\hat{\underline{x}}$. This can be done at the last check node iteration by convolving all the incoming variable node messages at each check node, without omitting any message. These estimates can be regarded as the pdf estimations of the elements of $\underline{b}$, without restricting them to be integers. Denote the estimated pdf for b_m by $f_{b_m}(x)$. Then, we can estimate b_m by $\hat{b}_m = arg\max_{j \in \mathbb{Z}} f_{b_m}(j)$.

We have finished developing the iterative algorithm. It can be easily seen that the message passing formulation of Section III-A actually implements this algorithm.

APPENDIX IV
ASYMPTOTIC BEHAVIOR OF THE VARIANCES RECURSION

A. Proof of Lemma 3 and Lemma 4

We shall now derive the basic iterative equations that relate the variances at iteration $t+1$ to the variances at iteration t for a Latin square LDLC with dimension n, degree d and generating sequence $h_1 \geq h_2 \geq \cdots \geq h_d > 0$.

Each iteration, every check node generates d output messages, one for each variable node that is connected to it, where the weights of these d connections are $\pm h_1, \pm h_2, \ldots, \pm h_d$. For each such output message, the check node convolves $d-1$ expanded variable node pdf messages, and then stretches and periodically extends the result. For a specific check node, denote the variance of the variable node message that arrives along an

edge with weight $\pm h_j$ by $V_j^{(t)}, j = 1, 2, \ldots, d$. Denote the variance of the message that is sent back to a variable node along an edge with weight $\pm h_j$ by $\tilde{V}_j^{(t)}$. From (2), (3), we get

$$\tilde{V}_j^{(t)} = \frac{1}{h_j^2} \sum_{\substack{i=1 \\ i \neq j}}^{d} h_i^2 V_i^{(t)}. \tag{38}$$

Then, each variable node generates d messages, one for each check node that is connected to it, where the weights of these d connections are $\pm h_1, \pm h_2, \ldots, \pm h_d$. For each such output message, the variable node generates the product of $d-1$ check node messages and the channel noise pdf. For a specific variable node, denote the variance of the message that is sent back to a check node along an edge with weight $\pm h_j$ by $V_j^{(t+1)}$ (this is the final variance of the iteration). From claim 2, we then get

$$\frac{1}{V_j^{(t+1)}} = \sum_{\substack{i=1 \\ i \neq j}}^{d} \frac{1}{\tilde{V}_i^{(t)}} + \frac{1}{\sigma^2}. \tag{39}$$

From symmetry considerations, it can be seen that all messages that are sent along edges with the same absolute value of their weight will have the same variance, since the same variance update occurs for all these messages (both for check node messages and variable node messages). Therefore, the d variance values $V_1^{(t)}, V_2^{(t)}, \ldots, V_d^{(t)}$ are the same for all variable nodes, where $V_l^{(t)}$ is the variance of the message that is sent along an edge with weight $\pm h_l$. This completes the proof of Lemma 3.

Using this symmetry, we can now derive the recursive update of the variance values $V_1^{(t)}, V_2^{(t)}, \ldots, V_d^{(t)}$. Substituting (38) in (39), we get

$$\frac{1}{V_i^{(t+1)}} = \frac{1}{\sigma^2} + \sum_{\substack{m=1 \\ m \neq i}}^{d} \frac{h_m^2}{\sum_{\substack{j=1 \\ j \neq m}}^{d} h_j^2 V_j^{(t)}} \tag{40}$$

for $i = 1, 2, \ldots, d$, which completes the proof of Lemma 4.

B. Proof of Theorem 1

We would like to analyze the convergence of the nonlinear recursion (4) for the variances $V_1^{(t)}, V_2^{(t)}, \ldots, V_d^{(t)}$. This recursion is illustrated in (5) for the case $d = 3$. It is assumed that $\alpha < 1$, where $\alpha = \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$. Define another set of variables $U_1^{(t)}, U_2^{(t)}, \ldots, U_d^{(t)}$, which obey the following recursion. The recursion for the first variable is

$$\frac{1}{U_1^{(t+1)}} = \frac{1}{\sigma^2} + \sum_{m=2}^{d} \frac{h_m^2}{h_1^2 U_1^{(t)}} \tag{41}$$

where for $i = 2, 3, \ldots, d$ the recursion is:

$$\frac{1}{U_i^{(t+1)}} = \frac{h_1^2}{\sum_{j=2}^{d} h_j^2 U_j^{(t)}}$$

with initial conditions $U_1^{(0)} = U_2^{(0)} = \cdots = U_d^{(0)} = \sigma^2$.

It can be seen that (41) can be regarded as the approximation of (4) under the assumptions that $V_i^{(t)} \ll V_1^{(t)}$ and $V_i^{(t)} \ll \sigma^2$ for $i = 2, 3, \ldots, d$.

For illustration, the new recursion for the case $d = 3$ is

$$\begin{aligned} \frac{1}{U_1^{(t+1)}} &= \frac{h_2^2}{h_1^2 U_1^{(t)}} + \frac{h_3^2}{h_1^2 U_1^{(t)}} + \frac{1}{\sigma^2} \\ \frac{1}{U_2^{(t+1)}} &= \frac{h_1^2}{h_2^2 U_2^{(t)} + h_3^2 U_3^{(t)}} \\ \frac{1}{U_3^{(t+1)}} &= \frac{h_1^2}{h_2^2 U_2^{(t)} + h_3^2 U_3^{(t)}}. \end{aligned} \tag{42}$$

It can be seen that in the new recursion, $U_1^{(t)}$ obeys a recursion that is independent of the other variables. From (41), this recursion can be written as $\frac{1}{U_1^{(t+1)}} = \frac{1}{\sigma^2} + \frac{\alpha}{U_1^{(t)}}$, with initial condition $U_1^{(0)} = \sigma^2$. Since $\alpha < 1$, this is a stable linear recursion for $\frac{1}{U_1^{(t)}}$, which can be solved to get $U_1^{(t)} = \sigma^2(1-\alpha)\frac{1}{1-\alpha^{t+1}}$.

For the other variables, it can be seen that all have the same right hand side in the recursion (41). Since all are initialized with the same value, it follows that $U_2^{(t)} = U_3^{(t)} = \cdots = U_d^{(t)}$ for all $t \geq 0$. Substituting back in (41), we get the recursion $U_2^{(t+1)} = \alpha U_2^{(t)}$, with initial condition $U_2^{(0)} = \sigma^2$. Since $\alpha < 1$, this is a stable linear recursion for $U_2^{(t)}$, which can be solved to get $U_2^{(t)} = \sigma^2 \alpha^t$.

We found an analytic solution for the variables $U_i^{(t)}$. However, we are interested in the variances $V_i^{(t)}$. The following claim relates the two sets of variables.

Claim 3: For every $t \geq 0$, the first variables of the two sets are related by $V_1^{(t)} \geq U_1^{(t)}$, where for $i = 2, 3, \ldots, d$ we have $V_i^{(t)} \leq U_i^{(t)}$.

Proof: By induction: the initialization of the two sets of variables obviously satisfies the required relations. Assume now that the relations are satisfied for iteration t, i.e., $V_1^{(t)} \geq U_1^{(t)}$ and for $i = 2, 3, \ldots, d$, $V_i^{(t)} \leq U_i^{(t)}$. If we now compare the right-hand side of the update recursion for $\frac{1}{V_1^{(t+1)}}$ to that of $\frac{1}{U_1^{(t+1)}}$ (i.e., (4) to (41)), then the right-hand side for $\frac{1}{V_1^{(t+1)}}$ is smaller, because it has additional positive terms in the denominators, where the common terms in the denominators are larger according to the induction assumption. Therefore, $V_1^{(t+1)} \geq U_1^{(t+1)}$, as required. In the same manner, if we compare the right-hand side of the update recursion for $\frac{1}{V_i^{(t+1)}}$ to that of $\frac{1}{U_i^{(t+1)}}$ for $i \geq 2$, then the right-hand side for $\frac{1}{V_i^{(t+1)}}$ is larger, because it has additional positive terms, where the common terms are also larger since their denominators are smaller due to the induction assumption. Therefore, $V_i^{(t+1)} \leq U_i^{(t+1)}$ for $i = 2, 3, \ldots, d$, as required. □

Using claim 3 and the analytic results for $U_i^{(t)}$, we now have

$$V_1^{(t)} \geq U_1^{(t)} = \sigma^2(1-\alpha)\frac{1}{1-\alpha^{t+1}} \geq \sigma^2(1-\alpha) \tag{43}$$

where for $i = 2, 3, \ldots, d$ we have:

$$V_i^{(t)} \leq U_i^{(t)} = \sigma^2 \alpha^t. \tag{44}$$

We have shown that the first variance is lower bounded by a positive nonzero constant where the other variances are upper bounded by a term that decays exponentially to zero. Therefore, for large t we have $V_i^{(t)} \ll V_1^{(t)}$ and $V_i^{(t)} \ll \sigma^2$ for $i = 2,3,\ldots,d$. It then follows that for large t the variances approximately obey the recursion (41), which was built from the actual variance recursion (4) under these assumptions. Therefore, for $i = 2,3,\ldots,d$ the variances are not only upper bounded by an exponentially decaying term, but actually approach such a term, where the first variance actually approaches the constant $\sigma^2(1-\alpha)$ in an exponential rate. This completes the proof of Theorem 1.

Note that the above analysis only applies if $\alpha < 1$. To illustrate the behavior for $\alpha \geq 1$, consider the simple case of $h_1 = h_2 = \cdots = h_d$. From (4), (5) it can be seen that for this case, if $V_i^{(0)}$ is independent of i, then $V_i^{(t)}$ is independent of i for every $t > 0$, since all the elements will follow the same recursive equations. Substituting this result in the first equation, we get the single variable recursion $\frac{1}{V_i^{(t+1)}} = \frac{1}{V_i^{(t)}} + \frac{1}{\sigma^2}$ with initialization $V_i^{(0)} = \sigma^2$. This recursion is easily solved to get $\frac{1}{V_i^{(t)}} = \frac{t+1}{\sigma^2}$ or $V_i^{(t)} = \frac{\sigma^2}{t+1}$. It can be seen that all the variances converge to zero, but with slow convergence rate of $o(1/t)$.

Appendix V
Asymptotic Behavior of the Mean Values Recursion

A. Proof of Lemma 5 and Lemma 6 (Mean of Narrow Messages)

Assume a Latin square LDLC with dimension n and degree d. We shall now examine the effect of the calculations in the check nodes and variable nodes on the mean values and derive the resulting recursion. Every iteration, each check node generates d output messages, one for each variable node that connects to it, where the weights of these d connections are $\pm h_1, \pm h_2, \ldots, \pm h_d$. For each such output message, the check node convolves $d-1$ expanded variable node pdf messages, and then stretches and periodically extends the result. We shall concentrate on the nd consistent Gaussians that relate to the same integer vector $\underline{b}$ (one Gaussian in each message), and analyze them jointly. For convenience, we shall refer to the mean value of the relevant consistent Gaussian as the mean of the message.

Consider now a specific check node. Denote the mean value of the variable node message that arrives at iteration t along the edge with weight $\pm h_j$ by $m_j^{(t)}$, $j = 1,2,\ldots,d$. Denote the mean value of the message that is sent back to a variable node along an edge with weight $\pm h_j$ by $\tilde{m}_j^{(t)}$. From (2), (3) and claim 1, we get

$$\tilde{m}_j^{(t)} = \frac{1}{h_j}\left(b_k - \sum_{\substack{i=1 \\ i\neq j}}^{d} h_i m_i^{(t)}\right) \tag{45}$$

where b_k is the appropriate element of $\underline{b}$ that is related to this specific check equation, which is the only relevant index in the infinite sum of the periodic extension step (3). Note that the check node operation is equivalent to extracting the value of m_j from the check equation $\sum_{i=1}^{d} h_i m_i = b_k$, assuming all the other m_i are known. Note also that the coefficients h_j should have a random sign. To keep notations simple, we assume that h_j already includes the random sign. Later, when several equations will be combined together, we should take it into account.

Then, each variable node generates d messages, one for each check node that is connected to it, where the weights of these d connections are $\pm h_1, \pm h_2, \ldots, \pm h_d$. For each such output message, the variable node generates the product of $d-1$ check node messages and the channel noise pdf. For a specific variable node, denote the mean value of the message that arrives from a check node along an edge with weight $\pm h_j$ by $\tilde{m}_j^{(t)}$, and the appropriate variance by $\tilde{V}_j^{(t)}$. The mean value of the message that is sent back to a check node along an edge with weight $\pm h_j$ is $m_j^{(t+1)}$, the final mean value of the iteration. From claim 2, we then get

$$m_j^{(t+1)} = \frac{y_k/\sigma^2 + \sum_{\substack{i=1 \\ i\neq j}}^{d} \tilde{m}_i^{(t)}/\tilde{V}_i^{(t)}}{1/\sigma^2 + \sum_{\substack{i=1 \\ i\neq j}}^{d} 1/\tilde{V}_i^{(t)}} \tag{46}$$

where y_k is the channel observation for the variable node and σ^2 is the noise variance. Note that $\tilde{m}_i^{(t)}$, $i = 1,2,\ldots,d$ in (46) are the mean values of check node messages that arrive to the same variable node from different check nodes, where in (45) they define the mean values of check node messages that leave the same check node. However, it is beneficial to keep the notations simple, and we shall take special care when (46) and (45) are combined.

It can be seen that the convergence of the mean values is coupled to the convergence of the variances (unlike the recursion of the variances which was autonomous). However, as the iterations go on, this coupling disappears. To see that, recall from Theorem 1 that for each check node, the variance of the variable node message that comes along an edge with weight $\pm h_1$ approaches a finite value, where the variance of all the other messages approaches zero exponentially. According to (38), the variance of the check node message is a weighted sum of the variances of the incoming variable node messages. Therefore, the variance of the check node message that goes along an edge with weight $\pm h_1$ will approach zero, since the weighted sum involves only zero-approaching variances. All the other messages will have finite variance, since the weighted sum involves the nonzero-approaching variance. To summarize, each variable node sends (and each check node receives) $d-1$ "narrow" messages and a single "wide" message. Each check node sends (and each variable node receives) $d-1$ "wide" messages and a single "narrow" message, where the narrow message is sent along the edge from which the wide message was received (the edge with weight $\pm h_1$).

We shall now concentrate on the case where the variable node generates a narrow message. Then, the sum in the numerator of (46) has a single term for which $\tilde{V}_i^{(t)} \to 0$, which corresponds to $i = 1$. The same is true for the sum in the denominator. Therefore, for large t, all the other terms will become negligible and we get

$$m_j^{(t+1)} \approx \tilde{m}_1^{(t)} \tag{47}$$

where $\tilde{m}_1^{(t)}$ is the mean of the message that comes from the edge with weight h_1, i.e., the narrow check node message. As discussed above, $d-1$ of the d variable node messages that leave the same variable node are narrow. From (47) it comes out that for large t, all these $d-1$ narrow messages will have the same mean value. This completes the proof of Lemma 5.

Now, combining (45) and (47) (where the indices are arranged again, as discussed above), we get

$$m_{l_1}^{(t+1)} \approx \frac{1}{h_1}\left(b_k - \sum_{i=2}^{d} h_i m_{l_i}^{(t)}\right) \tag{48}$$

where $l_i, i = 1, 2, \ldots, d$ are the variable nodes that take place in the check equation for which variable node l_1 appears with coefficient $\pm h_1$. b_k is the element of $\underline{b}$ that is related to this check equation. $m_{l_1}^{(t+1)}$ denotes the mean value of the $d-1$ narrow messages that leave variable node l_1 at iteration $t+1$. $m_{l_i}^{(t)}$ is the mean value of the narrow messages that were generated at variable node l_i at iteration t. Only narrow messages are involved in (48), because the right hand side of (47) is the mean value of the *narrow* check node message that arrived to variable node l_1, which results from the convolution of $d-1$ *narrow* variable node messages. Therefore, for large t, the mean values of the narrow messages are decoupled from the mean values of the wide messages (and also from the variances), and they obey an autonomous recursion.

The mean values of the narrow messages at iteration t can be arranged in an n-element column vector $\underline{m}^{(t)}$ (one mean value for each variable node). We would like to show that the mean values converge to the coordinates of the lattice point $\underline{x} = \boldsymbol{G}\underline{b}$. Therefore, it is useful to define the error vector $\underline{e}^{(t)} \triangleq \underline{m}^{(t)} - \underline{x}$. Since $\boldsymbol{H}\underline{x} = \underline{b}$, we can write (using the same notations as (48))

$$x_{l_1} = \frac{1}{h_1}\left(b_k - \sum_{i=2}^{d} h_i x_{l_i}\right). \tag{49}$$

Subtracting (49) from (48), we get

$$e_{l_1}^{(t+1)} \approx -\frac{1}{h_1}\sum_{i=2}^{d} h_i e_{l_i}^{(t)}. \tag{50}$$

Or, in vector notation

$$\underline{e}^{(t+1)} \approx -\tilde{\boldsymbol{H}} \cdot \underline{e}^{(t)} \tag{51}$$

where $\tilde{\boldsymbol{H}}$ is derived from $\boldsymbol{H}$ by permuting the rows such that the $\pm h_1$ elements will be placed on the diagonal, dividing each row by the appropriate diagonal element (h_1 or $-h_1$), and then nullifying the diagonal. Note that in order to simplify the notations, we embedded the sign of $\pm h_j$ in h_j and did not write it implicitly. However, the definition of $\tilde{\boldsymbol{H}}$ solves this ambiguity. This completes the proof of Lemma 6.

B. Proof of Lemma 7 (Mean of Wide Messages)

Recall that each check node receives $d-1$ narrow messages and a single wide message. The wide message comes along the edge with weight $\pm h_1$. Denote the appropriate lattice point by $\underline{x} = \boldsymbol{G}\underline{b}$, and assume that the Gaussians of the narrow variable node messages have already converged to impulses at the corresponding lattice point coordinates (Theorem 2). We can then substitute in (45) $m_i^{(t)} = x_i$ for $i \geq 2$. The mean value of the (wide) message that is returned along the edge with weight $\pm h_j$ ($j \neq 1$) is

$$\begin{aligned}\tilde{m}_j^{(t)} &= \frac{1}{h_j}\left(b_k - \sum_{\substack{i=2 \\ i\neq j}}^{d} h_i x_i - h_1 m_1^{(t)}\right) \\ &= \frac{1}{h_j}\left(h_1 x_1 + h_j x_j - h_1 m_1^{(t)}\right) = x_j + \frac{h_1}{h_j}\left(x_1 - m_1^{(t)}\right).\end{aligned} \tag{52}$$

As in the previous section, for convenience of notations we embed the sign of $\pm h_j$ in h_j itself. The sign ambiguity will be resolved later.

The meaning of (52) is that the returned mean value is the desired lattice coordinate plus an error term that is proportional to the error in the incoming wide message. From (38), assuming that the variance of the incoming wide message has already converged to its steady-state value $\sigma^2(1-\alpha)$ and the variance of the incoming narrow messages has already converged to zero, the variance of this check node message will be

$$\tilde{V}_j^{(t)} = \frac{h_1^2}{h_j^2}\sigma^2(1-\alpha) \tag{53}$$

where $\alpha = \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$. Now, each variable node receives $d-1$ wide messages and a single narrow message. The mean values of the wide messages are according to (52) and the variances are according to (53). The single wide message that this variable node generates results from the $d-1$ input wide messages and it is sent along the edge with weight $\pm h_1$. From (46), the wide mean value generated at variable node k will then be

$$m_k^{(t+1)} = \frac{y_k/\sigma^2 + \sum_{j=2}^{d}\left(x_k + \frac{h_1}{h_j}\left(x_{p(k,j)} - m_{p(k,j)}^{(t)}\right)\right)\frac{h_j^2}{h_1^2\sigma^2(1-\alpha)}}{1/\sigma^2 + \sum_{j=2}^{d}\frac{h_j^2}{h_1^2\sigma^2(1-\alpha)}}. \tag{54}$$

Note that the x_1 and m_1 terms of (52) were replaced by $x_{p(k,j)}$ and $m_{p(k,j)}$, respectively, since for convenience of notations we denoted by m_1 the mean of the message that came to a check node along the edge with weight $\pm h_1$. For substitution in (46) we need to know the exact variable node index that this edge came from. Therefore, $p(k,j)$ denotes the index of the variable node that takes place with coefficient $\pm h_1$ in the check equation where x_k takes place with coefficient $\pm h_j$.

Rearranging terms, we then get

$$\begin{aligned}m_k^{(t+1)} &= \frac{y_k(1-\alpha) + x_k\cdot\alpha + \sum_{j=2}^{d}\frac{h_j}{h_1}\left(x_{p(k,j)} - m_{p(k,j)}^{(t)}\right)}{(1-\alpha)+\alpha} \\ &= y_k + \alpha(x_k - y_k) + \frac{1}{h_1}\sum_{j=2}^{d} h_j(x_{p(k,j)} - m_{p(k,j)}^{(t)}).\end{aligned} \tag{55}$$

Denote now the wide message mean value error by $e_k^{(t)} \triangleq m_k^{(t)} - x_k$ (where $\underline{x} = \boldsymbol{G}\underline{b}$ is the lattice point that corresponds to $\underline{b}$). Denote by $\underline{q}$ the difference vector between

$\underline{x}$ and the noisy observation $\underline{y}$, i.e., $\underline{q} \triangleq \underline{y} - \underline{x}$. Note that if $\underline{b}$ corresponds to the correct lattice point that was transmitted, $\underline{q}$ equals the channel noise vector $\underline{w}$. Subtracting x_k from both sides of (55), we finally get

$$e_k^{(t+1)} = q_k(1-\alpha) - \frac{1}{h_1}\sum_{j=2}^{d} h_j e_{p(k,j)}^{(t)}. \tag{56}$$

If we now arrange all the errors in a single column vector $\underline{e}$, we can write

$$\underline{e}^{(t+1)} = -\boldsymbol{F}\cdot\underline{e}^{(t)} + (1-\alpha)\underline{q} \tag{57}$$

where $\boldsymbol{F}$ is an $n \times n$ matrix defined by

$$F_{k,l} = \begin{cases} \frac{H_{r,k}}{H_{r,l}}, & \text{if } k \neq l \text{ and there exist a row } r \text{ of } H \\ & \text{for which } |H_{r,l}| = h_1 \text{ and } H_{r,k} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{58}$$

$\boldsymbol{F}$ is well defined, since for a given l there can be at most a single row of $\boldsymbol{H}$ for which $|H_{r,l}| = h_1$ (note that $\alpha < 1$ implies that h_1 is different from all the other elements of the generating sequence).

As discussed above, we embedded the sign in h_i for convenience of notations, but when several equations are combined the correct signs should be used. It can be seen that using the notations of (57) resolves the correct signs of the h_i elements. This completes the proof of Lemma 7.

An alternative way to construct $\boldsymbol{F}$ from $\boldsymbol{H}$ is as follows. To construct the kth row of $\boldsymbol{F}$, denote by r_i, $i = 1, 2, \ldots, d$, the index of the element in the kth column of $\boldsymbol{H}$ with value h_i (i.e., $|H_{r_i,k}| = h_i$). Denote by l_i, $i = 1, 2, \ldots, d$, the index of the element in the r_ith row of $\boldsymbol{H}$ with value h_1 (i.e., $|H_{r_i,l_i}| = h_1$). The kth row of $\boldsymbol{F}$ will be all zeros except for the $d-1$ elements l_i, $i = 2, 3, \ldots, d$, where $F_{k,l_i} = \frac{H_{r_i,k}}{H_{r_i,l_i}}$.

APPENDIX VI
ASYMPTOTIC BEHAVIOR OF THE AMPLITUDES RECURSION

A. Proof of Lemma 9

From (10), $a_i^{(t)}$ is clearly nonnegative. From Sections IV-B, IV-C (and the appropriate appendices) it comes out that for consistent Gaussians, the mean values and variances of the messages have a finite bounded value and converge to a finite steady-state value. The excitation term $a_i^{(t)}$ depends on these mean values and variances according to (10), so it is also finite and bounded, and it converges to a steady-state value, where caution should be taken for the case of a zero approaching variance. Note that at most a single variance in (10) may approach zero (as explained in Section IV-B, a single narrow check node message is used for the generation of narrow variable node messages, and only wide check node messages are used for the generation of wide variable node messages). The zero approaching variance corresponds to the message that arrives along an edge with weight $\pm h_1$, so assume that $\tilde{V}_{k,1}^{(t)}$ approaches zero and all other variances approach a nonzero value. Then, $\hat{V}_{k,i}^{(t)}$ also approaches zero and we have to show that the term $\frac{\hat{V}_{k,i}^{(t)}}{\tilde{V}_{k,1}^{(t)}}$, which is a quotient of zero approaching terms, approaches a finite value. Substituting for $\hat{V}_{k,i}^{(t)}$, we get

$$\lim_{\tilde{V}_{k,1}^{(t)}\to 0} \frac{\hat{V}_{k,i}^{(t)}}{\tilde{V}_{k,1}^{(t)}} = \lim_{\tilde{V}_{k,1}^{(t)}\to 0} \frac{1}{\tilde{V}_{k,1}^{(t)}} \left(\frac{1}{\sigma^2} + \sum_{\substack{j=1 \\ j\neq i}}^{d} \frac{1}{\tilde{V}_{k,j}^{(t)}} \right)^{-1}$$
$$= \lim_{\tilde{V}_{k,1}^{(t)}\to 0} \left(\frac{\tilde{V}_{k,1}^{(t)}}{\sigma^2} + 1 + \sum_{\substack{j=2 \\ j\neq i}}^{d} \frac{\tilde{V}_{k,1}^{(t)}}{\tilde{V}_{k,j}^{(t)}} \right)^{-1} = 1. \tag{59}$$

Therefore, $a_i^{(t)}$ converges to a finite steady-state value, and has a finite value for every i and t. This completes the first part of the proof.

We would now like to show that $\lim_{t\to\infty} \sum_{i=1}^{nd} a_i^{(t)}$ can be expressed in the form $\frac{1}{2\sigma^2}(\boldsymbol{G}\underline{b} - \underline{y})^T \boldsymbol{W} (\boldsymbol{G}\underline{b} - \underline{y})$. Every variable node sends $d-1$ narrow messages and a single wide message. We shall start by calculating $a_i^{(t)}$ that corresponds to a narrow message. For this case, $d-1$ check node messages take place in the sums of (10), from which a single message is narrow and $d-2$ are wide. The narrow message arrives along the edge with weight $\pm h_1$, and has variance $\tilde{V}_{k,1}^{(t)} \to 0$. Substituting in (10), and using (59), we get

$$a_{(k-1)d+i}^{(t)} \to \frac{1}{2} \left(\sum_{\substack{j=2 \\ j\neq i}}^{d} \frac{\left(\tilde{m}_{k,1}^{(t)} - \tilde{m}_{k,j}^{(t)}\right)^2}{\tilde{V}_{k,j}^{(t)}} + \frac{\left(\tilde{m}_{k,1}^{(t)} - y_k\right)^2}{\sigma^2} \right). \tag{60}$$

Denote $\underline{x} = \boldsymbol{G}\underline{b}$. The mean values of the narrow check node messages converge to the appropriate lattice point coordinates, i.e., $\tilde{m}_{k,1}^{(t)} \to x_k$. From Theorem 3, the mean value of the wide variable node message that originates from variable node k converges to $x_k + e_k$, where $\underline{e}$ denotes the vector of error terms. The mean value of a wide check node message that arrives to node k along an edge with weight $\pm h_j$ can be seen to approach $\tilde{m}_{k,j}^{(t)} = x_k - \frac{h_1}{h_j} e_{p(k,j)}$, where $p(k,j)$ denotes the index of the variable node that takes place with coefficient $\pm h_1$ in the check equation where x_k takes place with coefficient $\pm h_j$. For convenience of notations, we shall assume that h_j already includes the sign (this sign ambiguity will be resolved later). The variance of the wide variable node messages converges to $\sigma^2(1-\alpha)$, so the variance of the wide check node message that arrives to node k along an edge with weight $\pm h_j$ can be seen to approach $\tilde{V}_{k,j}^{(t)} \to \frac{h_j^2}{h_j^2}\sigma^2(1-\alpha)$. Substituting in (60), and denoting $\underline{q} = \underline{y} - \underline{x}$, we get

$$a_{(k-1)d+i}^{(t)} \to \frac{1}{2} \left(\sum_{\substack{j=2 \\ j\neq i}}^{d} \frac{\left(\frac{h_1}{h_j} e_{p(k,j)}\right)^2}{\frac{h_j^2}{h_j^2}\sigma^2(1-\alpha)} + \frac{(x_k - y_k)^2}{\sigma^2} \right)$$
$$= \frac{1}{2\sigma^2} \left[\left(\frac{1}{1-\alpha} \sum_{\substack{j=2 \\ j\neq i}}^{d} e_{p(k,j)}^2 \right) + q_k^2 \right]. \tag{61}$$

Summing over all the narrow messages that leave variable node k, we get

$$\sum_{i=2}^{d} a_{(k-1)d+i}^{(t)} \to \frac{1}{2\sigma^2}\left[\left(\frac{d-2}{1-\alpha}\sum_{j=2}^{d} e_{p(k,j)}^2\right) + (d-1)q_k^2\right]. \tag{62}$$

To complete the calculation of the contribution of node k to the excitation term, we still have to calculate $a_i^{(t)}$ that corresponds to a wide message. Substituting $\tilde{m}_{k,j}^{(t)} \to x_k - \frac{h_1}{h_j}e_{p(k,j)}$, $\tilde{V}_{k,j}^{(t)} \to \frac{h_1^2}{h_j^2}\sigma^2(1-\alpha)$, $\hat{V}_{k,1}^{(t)} \to \sigma^2(1-\alpha)$ in (10), we get

$$a_{(k-1)d+1}^{(t)} \to \frac{1}{2}\sum_{l=2}^{d}\sum_{j=l+1}^{d} \frac{\left(\frac{h_1}{h_l}e_{p(k,l)} - \frac{h_1}{h_j}e_{p(k,j)}\right)^2}{\frac{h_1^2}{h_l^2}\cdot\frac{h_1^2}{h_j^2}\sigma^2(1-\alpha)} + \frac{1}{2}\sum_{l=2}^{d}\frac{\left(x_k - \frac{h_1}{h_l}e_{p(k,l)} - y_k\right)^2}{\frac{h_1^2}{h_l^2}\sigma^2}. \tag{63}$$

Starting with the first term, we have

$$\begin{aligned}
&\sum_{l=2}^{d}\sum_{j=l+1}^{d}\frac{\left(\frac{h_1}{h_l}e_{p(k,l)}-\frac{h_1}{h_j}e_{p(k,j)}\right)^2}{\frac{h_1^2}{h_l^2}\cdot\frac{h_1^2}{h_j^2}}\\
&=\frac{1}{2}\sum_{l=2}^{d}\sum_{j=2}^{d}\left(\frac{h_j}{h_1}e_{p(k,l)}-\frac{h_l}{h_1}e_{p(k,j)}\right)^2\\
&=\frac{1}{2}\sum_{l=2}^{d}\sum_{j=2}^{d}\left(\frac{h_j^2}{h_1^2}e_{p(k,l)}^2+\frac{h_l^2}{h_1^2}e_{p(k,j)}^2-2\frac{h_j}{h_1}\frac{h_l}{h_1}e_{p(k,l)}e_{p(k,j)}\right)\\
&=\alpha\sum_{j=2}^{d}e_{p(k,j)}^2-\left(\sum_{j=2}^{d}\frac{h_j}{h_1}e_{p(k,j)}\right)^2\\
&=\alpha\sum_{j=2}^{d}e_{p(k,j)}^2-(\boldsymbol{F}\cdot\underline{e})_k^2
\end{aligned}\tag{64}$$

where $\boldsymbol{F}$ is defined in Theorem 3 and $(\boldsymbol{F}\cdot\underline{e})_k$ denotes the kth element of the vector $(\boldsymbol{F}\cdot\underline{e})$. Note that using $\boldsymbol{F}$ solves the sign ambiguity that results from embedding the sign of $\pm h_j$ in h_j for convenience of notations, as discussed above. Turning now to the second term of (63)

$$\begin{aligned}
&\sum_{l=2}^{d}\frac{\left(x_k-\frac{h_1}{h_l}e_{p(k,l)}-y_k\right)^2}{\frac{h_1^2}{h_l^2}}\\
&=\sum_{l=2}^{d}\left(e_{p(k,l)}^2+\frac{h_l^2}{h_1^2}q_k^2+2q_ke_{p(k,l)}\frac{h_l}{h_1}\right)\\
&=\left(\sum_{l=2}^{d}e_{p(k,l)}^2\right)+\alpha q_k^2+2q_k\left(\boldsymbol{F}\cdot\underline{e}\right)_k\\
&=\left(\sum_{l=2}^{d}e_{p(k,l)}^2\right)+\alpha q_k^2+2q_k[(1-\alpha)q_k-e_k]\\
&=\left(\sum_{l=2}^{d}e_{p(k,l)}^2\right)+(2-\alpha)q_k^2-2q_ke_k
\end{aligned}\tag{65}$$

where we have substituted $\boldsymbol{F}\underline{e} \to (1-\alpha)\underline{q} - \underline{e}$, as comes out from Lemma 7. Again, using $\bar{\boldsymbol{F}}$ resolves the sign ambiguity of h_j, as discussed above.

Substituting (64) and (65) back in (63), summing the result with (62), and rearranging terms, the total contribution of variable node k to the asymptotic excitation sum term is

$$\sum_{i=1}^{d} a_{(k-1)d+i}^{(t)} \to \frac{d-1}{2\sigma^2(1-\alpha)}\sum_{j=2}^{d}e_{p(k,j)}^2 + \frac{d+1-\alpha}{2\sigma^2}q_k^2 - \frac{1}{2\sigma^2(1-\alpha)}(\boldsymbol{F}\underline{e})_k^2 - \frac{1}{\sigma^2}q_ke_k. \tag{66}$$

Summing over all the variable nodes, the total asymptotic excitation sum term is

$$\sum_{i=1}^{nd} a_i^{(t)} = \sum_{k=1}^{n}\sum_{i=1}^{d} a_{(k-1)d+i}^{(t)} \to \frac{(d-1)^2}{2\sigma^2(1-\alpha)}\left\|\underline{e}\right\|^2 + \frac{d+1-\alpha}{2\sigma^2}\left\|\underline{q}\right\|^2 - \frac{1}{2\sigma^2(1-\alpha)}\left\|\bar{\boldsymbol{F}}\underline{e}\right\|^2 - \frac{1}{\sigma^2}\underline{q}^T\underline{e}. \tag{67}$$

Substituting $\underline{e} = (1-\alpha)(\boldsymbol{I}+\boldsymbol{F})^{-1}\underline{q}$ (see Theorem 3), we finally get

$$\sum_{i=1}^{nd} a_i^{(t)} \to \frac{1}{2\sigma^2}\underline{q}^T\boldsymbol{W}\underline{q} \tag{68}$$

where

$$\boldsymbol{W} \triangleq (1-\alpha)(\boldsymbol{I}+\boldsymbol{F})^{-1^T}\left((d-1)^2\boldsymbol{I}-\boldsymbol{F}^T\boldsymbol{F}\right)(\boldsymbol{I}+\boldsymbol{F})^{-1} + (d+1-\alpha)\boldsymbol{I} - 2(1-\alpha)(\boldsymbol{I}+\boldsymbol{F})^{-1}. \tag{69}$$

From (10) it can be seen that $\sum_{i=1}^{nd} a_i^{(t)}$ is positive for every nonzero $\underline{q}$. Therefore, $\boldsymbol{W}$ is positive definite. This completes the second part of the proof.

Since $a_i^{(t)}$ is finite and bounded, there exists m_a such that $|a_i^{(t)}| \le m_a$ for all $1 \le i \le nd$ and $t > 0$. We then have

$$\sum_{j=0}^{\infty}\frac{\sum_{i=1}^{nd}a_i^{(j)}}{(d-1)^{2j+2}} \le \sum_{j=0}^{\infty}\frac{nd\cdot m_a}{(d-1)^{2j+2}} = \frac{n\cdot m_a}{(d-2)}.$$

Therefore, for $d > 2$ the infinite sum will have a finite steady-state value. This completes the proof of Lemma 9.

Appendix VII
Generation of a Parity Check Matrix for LDLC

In the following pseudocode description, the i,j element of a matrix P is denoted by $P_{i,j}$ and the kth column of a matrix P is denoted by $P_{:,k}$.

Input: block length n, degree d,

nonzero elements $\{h_1, h_2, \ldots, h_d\}$.

Output: a Latin square LDLC parity check matrix $\boldsymbol{H}$

with generating sequence $\{h_1, h_2, \ldots, h_d\}$.

Initialization:

choose d random permutations on $\{1, 2, \ldots, n\}$.
Arrange the permutations in an $d \times n$ matrix P

such that each row holds a permutation.

$c = 1$; # column index

$loopless\_columns = 0$; # number of consecutive
columns without loops

```
# loop removal:
while loopless_columns < n
    changed_permutation = 0;
    if exists i ≠ j such that P_{i,c} = P_{j,c}
      # a 2-loop was found at column c
      changed_permutation = i;
    else
      # if there is no 2-loop, look for a 4-loop
      if exists c_0 ≠ c such that P_{:,c} and P_{:,c_0} have
        two or more common elements
        # a 4-loop was found at column c
        changed_permutation = line of P for which
        the first common element appears in column c;
      end
    end
    if changed_permutation ≠ 0
        # a permutation should be modified to
        # remove loop
        choose a random integer 1 ≤ i ≤ n;
        swap locations c and i in
        permutation changed_permutation;
        loopless_columns = 0;
      else
        # no loop was found at column c
        loopless_columns = loopless_columns + 1;
      end
      # increase column index
      c = c + 1;
      if c > n
        c = 1;

      end
end

# Finally, build H from the permutations
initialize H as an n × n zero matrix;
for i = 1 : n
      for j = 1 : d
        H_{P_{j,i},i} = h_j · random_sign;
      end
end
```

Appendix VIII
Reducing the Complexity of the FFT Calculations

FFT calculation can be made simpler by using the fact that the convolution is followed by the following steps: the convolution result $\tilde{p}_j(x)$ is stretched to $p_j(x) = \tilde{p}_j(-h_j x)$ and then periodically extended to $Q_j(x) = \sum_{i=-\infty}^{\infty} p_j\left(x - \frac{i}{h_j}\right)$ (see (3)). It can be seen that the stretching and periodic extension steps can be exchanged, and the convolution result $\tilde{p}_j(x)$ can be first periodically extended with period 1 to $\tilde{Q}_j(x) = \sum_{i=-\infty}^{\infty} \tilde{p}_j(x+i)$ and then stretched to $Q_j(x) = \tilde{Q}_j(-h_j x)$. Now, the infinite sum can be written as a convolution with a sequence of Dirac impulses:

$$\tilde{Q}_j(x) = \sum_{i=-\infty}^{\infty} \tilde{p}_j(x+i) = \tilde{p}_j(x) * \sum_{i=-\infty}^{\infty} \delta(x+i). \quad (70)$$

Therefore, the Fourier transform of $\tilde{Q}_j(x)$ will equal the Fourier transform of $\tilde{p}_j(x)$ multiplied by the Fourier transform of the impulse sequence, which is itself an impulse sequence. The FFT of $\tilde{Q}_j(x)$ will therefore have several nonzero values, separated by sequences of zeros. These nonzero values will equal the FFT of $\tilde{p}_j(x)$ after decimation. To ensure an integer decimation rate, we should choose the pdf resolution Δ such that an interval with range 1 (the period of $\tilde{Q}_j(x)$) will contain an integer number of samples, i.e., $1/\Delta$ should be an integer. Also, we should choose L (the number of samples in $\tilde{Q}_j(x)$) to correspond to a range which equals an integer, i.e., $D = L \cdot \Delta$ should be an integer. Then, we can calculate the (size L) FFT of $\tilde{p}_j(x)$ and then decimate by D. The result will give $1/\Delta$ samples which correspond to a single period (with range 1) of $\tilde{Q}_j(x)$.

However, instead of calculating an FFT of length L and immediately decimating, we can directly calculate the decimated FFT. Denote the expanded pdf at the convolution input by $\tilde{f}_k$, $k = 1, 2, \ldots, L$ (where the expanded pdf is zero padded to length L). To generate directly the decimated result, we can first calculate the (size D) FFT of each group of D samples which are generated by decimating $\tilde{f}_k$ by $L/D = 1/\Delta$. Then, the desired decimated result is the FFT (of size $1/\Delta$) of the sequence of first samples of each FFT of size D. However, the first sample of an FFT is simply the sum of its inputs. Therefore, we should only calculate the sequence (of length $1/\Delta$) $g_i = \sum_{k=0}^{D-1} \tilde{f}_{i+k/\Delta}$, $i = 1, 2, \ldots, 1/\Delta$ and then calculate the FFT (of length $1/\Delta$) of the result. This is done for all the expanded pdfs. Then, $d-1$ such results are multiplied, and an IFFT (of length $1/\Delta$) gives a single period of $\tilde{Q}_j(x)$.

With this method, instead of calculating d FFTs and d IFFTs of size larger than L, we calculate d FFTs and d IFFTs of size $L/D = 1/\Delta$.

In order to generate the final check node message, we should stretch $\tilde{Q}_j(x)$ to $Q_j(x) = \tilde{Q}_j(-h_j x)$. This can be done by interpolating a single period of $\tilde{Q}_j(x)$ using interpolation methods similar to those that were used in Section VI for expanding the variable node pdfs.

Acknowledgment

Support and interesting discussions with Ehud Weinstein are gratefully acknowledged. The authors also like to thank the anonymous reviewers for their thorough review and valuable comments.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379, 623–423, 656, Jul.–Oct. 1948.
[2] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, vol. 3, pt. 4, pp. 37–46, Mar. 1955.
[3] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, 1959.
[4] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Commun.*, 1993, pp. 1064–1070.
[7] R. de Buda, "The upper error bound of a new near-optimal code," *IEEE Trans. Inf. Theory*, vol. IT-21, pp. 441–445, Jul. 1975.
[8] R. de Buda, "Some optimal codes have structure," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 893–899, Aug. 1989.
[9] T. Linder, C. Schlegel, and K. Zeger, "Corrected proof of de Buda's theorem," *IEEE Trans. Inf. Theory*, pp. 1735–1737, Sep. 1993.
[10] H. A. Loeliger, "Averaging bounds for lattices and linear codes," *IEEE Trans. Inf. Theory*, vol. 43, pp. 1767–1773, Nov. 1997.
[11] R. Urbanke and B. Rimoldi, "Lattice codes can achieve capacity on the AWGN channel," *IEEE Trans. Inf. Theory*, pp. 273–278, Jan. 1998.
[12] U. Erez and R. Zamir, "Achieving 1/2 $\log(1 + \mathrm{SNR})$ on the AWGN channel with lattice encoding and decoding," *IEEE Trans. Inf. Theory*, vol. 50, pp. 2293–2314, Oct. 2004.
[13] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inf. Theory*, vol. IT-33, pp. 177–195, Mar. 1987.
[14] G. D. Forney Jr., "Coset codes–Part I: Introduction and geometrical classification," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1123–1151, Sep. 1988.
[15] M. R. Sadeghi, A. H. Banihashemi, and D. Panario, "Low-density parity-check lattices: Construction and decoding analysis," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4481–4495, Oct. 2006.
[16] O. Shalvi, N. Sommer, and M. Feder, "Signal codes," in *Proc. Inf. Theory Workshop*, 2003, pp. 332–336.
[17] O. Shalvi, N. Sommer, and M. Feder, *Signal Codes*, to be published.
[18] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Trans. Inf. Theory*, vol. 52, pp. 549–583, Feb. 2006.
[19] J. Hou, P. H. Siegel, L. B. Milstein, and H. D. Pfister, "Capacity approaching bandwidth efficient coded modulation schemes based on low density parity check codes," *IEEE Trans. Inf. Theory*, vol. 49, pp. 2141–2155, Sep. 2003.
[20] J. H. Conway and N. J. Sloane, *Sphere Packings, Lattices and Groups*. New York: Springer, 1988.
[21] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Trans. Inf. Theory*, vol. 40, pp. 409–417, Mar. 1994.
[22] N. Wiberg, "Codes and Decoding on General Graphs," Doctoral, Dep. Elec. Eng., Linköping Univ., Linköping, Sweden, 1996.
[23] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.
[24] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Second Printing ed. San Mateo, CA: Morgan Kaufmann, 1988.
[25] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, second ed. New York: McGraw-Hill, 1984.
[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. New York: Society for Industrial and Applied Mathematic (SIAM), 2003.
[27] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, pp. 2201–2214, Aug. 2002.
[28] N. Sommer, M. Feder, and O. Shalvi, "Closest point search in lattices using sequential decoding," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2005, pp. 1053–1057.
[29] H. S. Cronie, "Superposition coding for power and bandwidth efficient communication over the Gaussian channel," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2007.
[30] H. El Gamal, G. Caire, and M. Damen, "Lattice coding and decoding achieve the optimal diversity-multiplexing tradeoff of MIMO channels," *IEEE Trans. Inf. Theory*, vol. 50, pp. 968–985, Jun. 2004.