Low Density Lattice Codes

Naftali Sommer[*,**], Meir Feder[*], Ofir Shalvi[*]
[*]Department of Electrical Engineering - Systems
Tel-Aviv University, Tel-Aviv, Israel, Email: meir@eng.tau.ac.il
[**]Texas Instruments, Herzlia, Israel

***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., $\underline{x} = G\underline{b}$, where $H = G^{-1}$ is restricted to be sparse. The fact that H is sparse is utilized to develop a linear-time iterative decoding scheme which attains, as demonstrated by simulations, good error performance within ~ 0.5dB from capacity at block length of $n = 100,000$ symbols. The paper also discusses convergence results and implementation considerations.**

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 BCH and Reed-Solomon codes [4]). Later, capacity achieving schemes were found, which have special structures that allow efficient 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 analogue of linear codes. However, there was almost no further progress from that point. Specific lattice codes that were found were based on fixed dimensional classical lattices [16] or based on algebraic error correcting codes [13][14], but no significant effort was made in designing lattice codes directly in the Euclidean space or in finding specific capacity achieving lattice codes.

In [15], "signal codes" were introduced. These are lattice codes, designed directly in the Euclidean space, where the information sequence of integers i_n, $n = 1, 2, ...$ is encoded by convolving it with a fixed signal pattern g_n, $n = 1, 2, ...d$. Signal codes are clearly analogous to convolutional codes, and can work at the AWGN channel cutoff rate with simple sequential decoders, providing the first step toward finding capacity approaching lattice codes with practical decoders.

Inspired by LDPC 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. Unlike LDPC, in LDLC both the encoder and the channel use the same real algebra which is natural for the AWGN. This feature also simplifies the convergence analysis of the iterative decoder.

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 G, whose columns are the basis vectors, is called a generator matrix of the lattice. 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 the volume of the Voronoi cell is equal to $det(G)$. A lattice code of dimension n is defined by a (possibly shifted) lattice 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.

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 [17] 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 [17] to use the measure of constellation density, leading to a generalized definition of the AWGN capacity. When applied to lattices, the generalized capacity implies that there exists a lattice G of high enough dimension n, with $det(G) = 1$, that enables transmission with arbitrary small error probability, if and only if $\sigma^2 < \frac{1}{2\pi e}$. For the high SNR regime, 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.

In the rest of the 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 $det(G) = 1$, 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 $\boldsymbol{G}$ (throughout this paper we assume that $\boldsymbol{G}$ is square, but the results are easily extended to the non-square case). 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 $H \stackrel{\Delta}{=} G^{-1}$. 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 $\underline{s} \stackrel{\Delta}{=} frac\{\boldsymbol{H}\underline{y}\}$, 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.

C. Low Density Lattice Codes

We shall now turn to the definition of the codes proposed in this paper - low density lattice codes (LDLC).

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

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 3 (magic square LDLC): An n dimensional regular LDLC with degree d is called "magic 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.

For example, the matrix

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

is a parity check matrix of a magic square LDLC with lattice dimension $n = 6$ and degree $d = 3$. The nonzero values are $\{1, 0.8, 0.5\}$. This H should be further normalized by the constant $\sqrt[n]{|det(H)|}$ in order to have $|det(H)| = |det(G)| = 1$, 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 H). 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 $H_{i,j}$.

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 calculating the n-dimensional PDF of the whole vector $\underline{x}$, we shall calculate the n one-dimensional PDF's for each of the components x_k of this vector (conditioned on the whole observation vector $\underline{y}$). It can be shown that $f_{x_k|\underline{y}}(x_k|\underline{y})$ is a weighted sum of Dirac delta functions:

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

where C is 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.

The calculation of $f_{\underline{x}|\underline{y}}(\underline{x}|\underline{y})$ is involved since the components x_k are not independent random variables (RV's), because $\underline{x}$ is restricted to be a lattice point. Following [5] we use a "trick" - we assume that the x_k's are i.i.d with a properly chosen PDF $f^{(i.i.d)}_{x_k}(x_k)$. This PDF is chosen to be the marginal distribution of a vector distribution $f^{(i.i.d)}_{\underline{x}}(\underline{x})$, which is uniformly distributed over the shaping region. In addition, we add a condition that assures that only lattice points have nonzero probability. Specifically, define $\underline{s} \stackrel{\Delta}{=} \boldsymbol{H} \cdot \underline{x}$. Define further the set of integer valued vectors i_B as $i_B = \{\underline{i} | \underline{i} \in \mathbb{Z}^n, \boldsymbol{G}\underline{i} \in B\}$. Restricting $\underline{x}$ to be a lattice point inside the shaping region B is equivalent to restricting $\underline{s} \in i_B$. Therefore,

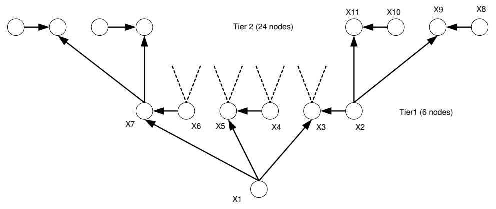

Fig. 1. Tier diagram

instead of calculating $f_{\underline{x}|\underline{y}}(\underline{x}|\underline{y})$ under the assumption that $\underline{x}$ is a lattice point in B, we can add the condition $\underline{s} \in i_B$ and assume that the x_k are i.i.d.

The derivation in [5] further imposed the tree assumption. Figure 1 shows the tier diagram of 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 algorithm is not guaranteed to converge to the exact solution (1) (see section IV).

The detailed derivation of the iterative decoder is omitted due to space constraints (the full derivation can be found in [18]). Below we present the final resulting algorithm. This iterative algorithm can also be explained by intuitive arguments, described 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, similarly to LDPC codes. The basic difference is that in LDPC codes the messages are scalar values (e.g. log likelihood ratio of a bit), where for LDLC the messages are real functions over the interval $[-\infty, \infty]$. As in LDPC, in each iteration the check nodes send messages to the variable nodes along the edges of the bipartite graph and visa versa. The messages sent by the check nodes are periodic extensions of PDF's. The messages sent by the variable nodes are PDF's.

LDLC iterative decoding algorithm:

Denote the variable nodes by $x_1, x_2, ..., x_n$ and the check nodes by $c_1, c_2, ...c_n$.

- *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}, $l = 1, 2...r$ are the variable nodes that are connected to this check node (and r is the appropriate row degree of H). Denote by $f_l(x)$, $l = 1, 2...r$, the message that was sent to this check node by variable node x_{m_l} in the previous iteration. The message that the check node transmits back to variable node x_{m_j} is calculated in three basic steps.

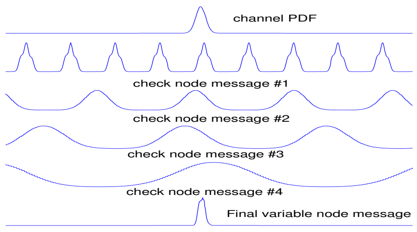

Fig. 2. Signals at variable node

1) *The convolution step* - all messages, except $f_j(x)$, are convolved (after expanding each $f_l(x)$ by h_l):

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

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 $1/|h_j|$:

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

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 $c_{m_1}, c_{m_2}, ...c_{m_e}$, where e is the appropriate column degree of H. Denote by $Q_l(x)$, $l = 1, 2, ...e$, the message that was sent from check node c_{m_l} to this variable node in the previous 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 $\underline{b}$. Each check node estimates the relevant integer, by calculating the convolution $\tilde{p}(x)$ as in (2), but this time without omitting any PDF. Then, the integer b_k is determined by $\hat{b_k} = arg \max_{j \in \mathbb{Z}} \tilde{p}(j)$.

The operation of the iterative algorithm can be intuitively explained as follows. The check node operation is equivalent to calculating the PDF of x_{m_j} from the PDF's of x_{m_i}, $i = 1, 2, ..., j-1, j+1, ...r$, 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 $x_{m_j} = \frac{1}{h_j}(integer - \sum_{\substack{l=1 \\ l \neq j}}^{r} h_l x_{m_l})$. Since the PDF of a sum of independent random variables is the convolution of the corresponding PDF's, equation (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.

This multiplication is illustrated in Figure 2, where it can be seen that each periodic signal has a different period, according to the relevant coefficient of 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.

IV. Convergence

A. The Gaussian Mixture Model

Interestingly, for LDLC we can come up with a convergence analysis. Here we provide an outline, while the detailed analysis is given in [18]. We start by introducing basic claims about Gaussian PDF's. We 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 $m_1, m_2, ..., m_n$ and variances $V_1, V_2, ..., V_n$, respectively, is a Gaussian with mean $m_1 + m_2 + ... + m_n$ and variance $V_1 + V_2 + ... + V_n$.

Claim 2 (product of n Gaussians): Let $G_{m_1,V_1}(x)$, $G_{m_2,V_2}(x)$,...,$G_{m_n,V_n}(x)$ be n Gaussians with mean values $m_1, m_2, ..., m_n$ and variances $V_1, V_2, ..., V_n$ respectively. Then, $\prod_{i=1}^{n} G_{m_i,V_i}(x) = \hat{A} \cdot G_{\hat{m},\hat{V}}(x)$, 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 $\hat{A} = \frac{1}{\sqrt{(2\pi)^{n-1} \hat{V}^{-1} \prod_{k=1}^{n} V_k}} \cdot e^{-\frac{\hat{V}}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(m_i - m_j)^2}{V_i \cdot V_j)}}$.

Now, the basic operations of the iterative decoder are convolution, multiplication and stretching. The fact that all these operations preserve the Gaussian nature of Gaussian inputs can be used to prove 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 $M(x) = \sum_{j=1}^{\infty} A_j G_{m_j,V_j}(x)$.

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.

Until this point we did not impose any restrictions on the LDLC. From now on, we shall restrict ourselves to magic square regular LDLC. Recall (see definition 3) that for such codes, the same d nonzero values (except for random signs and order) appear on each row or column of H. Denote these nonzero values by $h_1, h_2, ..., h_d$ and assume without loss of generality that $h_1 \geq h_2 \geq ... \geq h_d > 0$. 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: Variable node messages that are sent at the same iteration along edges with the same absolute value have the same variance.

Lemma 4: 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 $V_1^{(t)}, V_2^{(t)}, ..., V_d^{(t)}$ 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 $i = 1, 2, ...d$.

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

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

$$\frac{1}{V_2^{(t+1)}} = \frac{h_1^2}{h_2^2 V_2^{(t)} + h_3^2 V_3^{(t)}} + \frac{h_3^2}{h_1^2 V_1^{(t)} + h_2^2 V_2^{(t)}} + \frac{1}{\sigma^2}$$

$$\frac{1}{V_3^{(t+1)}} = \frac{h_1^2}{h_2^2 V_2^{(t)} + h_3^2 V_3^{(t)}} + \frac{h_2^2}{h_1^2 V_1^{(t)} + h_3^2 V_3^{(t)}} + \frac{1}{\sigma^2}$$

The lemmas above are used to prove the following theorem regarding the variance convergence in the general magic square LDLC case.

Theorem 1: For a magic square LDLC with degree d, define the sequence of nonzero elements that appear in every row and column (except for random signs and order) by $h_1, h_2, ..., h_d$, and assume without loss of generality that $h_1 \geq h_2 \geq ... \geq h_d > 0$. The asymptotic behavior of the variances depend on $\alpha \triangleq \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$ in the following manner:

1) If $\alpha < 1$, the first variance satisfies $\lim_{t\to\infty} V_1^{(t)} = \sigma^2(1-\alpha)$, where for $i = 2, 3..d$ we have $\lim_{t\to\infty} V_i^{(t)} = 0$ such that $0 < \lim_{t\to\infty} \frac{V_i^{(t)}}{\alpha^t} < \infty$.
2) If $\alpha \geq 1$, then for $i = 1, 2..., d$ we have $\lim_{t\to\infty} V_i^{(t)} = 0$, such that $0 < \lim_{t\to\infty} V_i^{(t)} \cdot t < \infty$.

It can be seen that by choosing $h_1^2 \leq \sum_{i=2}^{d} h_i^2$ we have all the variances approach zero, but in a slow rate of $1/t$. If we choose $h_1^2 > \sum_{i=2}^{d} h_i^2$ we have one variance that approaches a constant (and not zero). However, all the other variances approach zero 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 the slow $1/t$ convergence.

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 $1, 2, ...t$.

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 these Gaussians correspond to the lattice point $G\underline{b}$.

According to theorem 1, if we choose the nonzero values of H such that $\alpha < 1$, every variable node generates $d - 1$ messages with zero approaching variance 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. The following lemmas summarize the asymptotic behavior of the mean values of these consistent Gaussians for the narrow messages.

Lemma 5: Asymptotically, the mean values of the consistent Gaussians that relate to a given integer vector $\underline{b}$ are the same for all the narrow variable node messages of the same variable node.

Lemma 6: Denote the common mean value of the narrow messages of the i'th variable node at iteration t by $m_i^{(t)}$, and arrange all these mean values in a column vector $\underline{m}^{(t)}$ of dimension n. Then, for large t, $\underline{m}^{(t)}$ satisfies:

$$\underline{m}^{(t+1)} \approx -\tilde{H} \cdot \underline{m}^{(t)} + \frac{1}{h_1}\tilde{\underline{b}} \tag{6}$$

where $\tilde{H}$ is derived from 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. $\tilde{\underline{b}}$ is derived by flipping the sign of the elements of $\underline{b}$ for which the largest magnitude element of the appropriate row of H is $-h_1$, and then permuting with the same permutation that was used for the rows of H.

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: The mean values of the consistent Gaussians of the narrow variable node messages are assured to converge if and only if all the eigenvalues of $\tilde{H}$ have magnitude less than 1. When this condition is fulfilled, the steady state solution is $\underline{m}^{(\infty)} = G \cdot \underline{b}$.

We can see that the mean values of the consistent Gaussians converge to the coordinates of the appropriate lattice point. Interestingly, recursion (6) is equivalent to the Jacobi method for solving systems of sparse linear equations [19], where here we solve $H\underline{m} = \underline{b}$. Note that without adding random signs to the LDLC nonzero values, the all-ones vector will be an eigenvector of $\tilde{H}$ with eigenvalue $\frac{\sum_{i=2}^{d} h_i}{h_1}$, which may exceed 1.

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: Denote the mean value of the wide message of the i'th variable node at iteration t by $m_i^{(t)}$, and the appropriate error by $e_i^{(t)} \triangleq m_i^{(t)} - x_i$, where $\underline{x} = G\underline{b}$. Arrange all the error values in a column vector $\underline{e}^{(t)}$ of dimension n. Then, for large t, $\underline{e}^{(t)}$ satisfies:

$$\underline{e}^{(t+1)} \approx -F \cdot \underline{e}^{(t)} + (1 - \alpha)\underline{w} \tag{7}$$

where $\underline{w}$ is the channel noise vector, α is defined in theorem 1 and the matrix F can be constructed from H as follows. To construct the k'th row of F, denote by r_i, $i = 1, 2, ...d$, the index of the element in the k'th column of H with value h_i (i.e. $|H_{r_i,k}| = h_i$). Denote by l_i, $i = 1, 2, ...d$, the index of the element in the r_i'th row of H with value h_1 (i.e. $|H_{r_i,l_i}| = h_1$). The k'th row of F will then be all zeros, except for the $d - 1$ locations l_i, $i = 2, 3...d$, where $F_{k,l_i} = \frac{H_{r_i,k}}{H_{r_i,l_i}}$.
The conditions for convergence and steady state solution for the wide messages are brought in the following theorem.

Theorem 3: The recursion (7) is assured to converge if and only if all the eigenvalues of F have magnitude less than 1.

When this condition is fulfilled, the steady state solution is $\underline{e}^{(\infty)} = (1-\alpha)(I+F)^{-1}\underline{w}$.

Unlike the narrow messages, the mean values of the wide messages do not converge to the appropriate lattice point coordinates, since the error vector does not converge to zero. The steady state error depends on the channel noise vector and on the parameter α, and it decreases to zero as $\alpha \to 1$. Note that this error term should not be a problem because convergence of the narrow messages is sufficient for extracting the information. In principle, the wide messages can even diverge, as long as they remain wide. However, in a practical implementation where the PDF's have finite range, divergence of the wide messages should be avoided.

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 H, the mean values of all the narrow messages converge to the appropriate coordinate of the lattice point $G\underline{b}$. Under additional conditions on H, the mean values of the wide messages converge, but the steady state values contain an error term.

We considered consistent Gaussians which corresponds to a specific integer vector $\underline{b}$, but such a set of Gaussians will exist 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. However, the analysis of the amplitudes is more complex and was not yet finalized.

V. Code Design

We shall concentrate on magic 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 sequence $h_1, h_2, ...h_d$ for the magic square LDLC construction. 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 $\{1/2, 1/3, 1/5, 1/7, 1/11, 1/13, 1/17, ...\}$, i.e. the reciprocals of the smallest d prime numbers. Since the periods are $1/h_1, 1/h_2, ...1/h_d$, 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.

We shall now test the conditions on H that were mentioned in section IV. For these parameters we get $\alpha = 0.92$ and 0.87 for $d = 7$ and 5, respectively, which is a reasonable trade off. For $n \geq 1000$, the largest eigenvalue of $\tilde{H}$ has magnitude of $0.94 - 0.97$, almost independently of n and the choice of nonzero H locations. For $n = 100$ there were rare occasions where it exceeds 1, and these H matrices should be avoided.

Finally, we shall present a simple algorithm for constructing a parity check matrix for a magic 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 $h_1, h_2, ...h_d$. All the edges that have the same weight h_j form a permutation from the variable nodes to the check nodes (or visa 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.

VI. Implementation and Complexity

Each PDF should be approximated with a discrete vector with resolution Δ and 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.

The most computationally intensive part in the basic iteration are the convolutions at the check nodes, so it is natural to use FFT's. It can be shown that FFT size can be $1/\Delta$ instead of L (64 instead of 256 for the parameters above).

The stretching of PDF's is done using interpolation, that averages the neighboring points of the desired location. Multiplication of check node messages is preceded by widening the messages by 1 sample to each side. These operations inhibit impulses from disappearing due to finite resolution.

Most of the computational effort is invested in the d FFT's and d IFFT's (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 10^6.

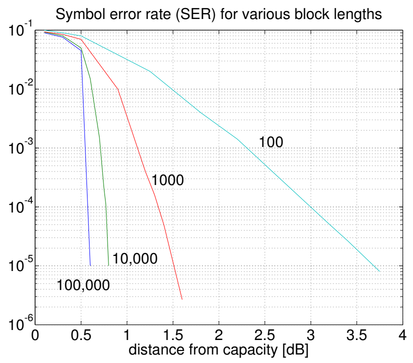

Fig. 3. Simulation results

VII. Simulation Results

Magic square LDLC with the generating sequence of section V 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 H was further normalized to get $\sqrt[n]{det(H)} = 1$. For all other n, normalizing the generating sequence such that the largest element has magnitude 1 also gave the desired determinant normalization. The H matrices were generated using the algorithm of section V. PDF resolution was set to $\Delta = 1/256$ with a total range of 4, i.e. $L = 1024$. 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).

According to Poltyrev's results (section II-A), shaping is not necessary and the all-zero codeword was used. Approaching channel capacity is equivalent to $\sigma^2 \rightarrow \frac{1}{2\pi e}$, so performance is measured in symbol error rate (SER), vs. the distance of the noise variance σ^2 from capacity (in dB). The results are shown in Figure 3. At SER of 10^{-5}, for $n = 100000$, 10000, 1000, 100 we can work as close as 0.6dB, 0.8dB, 1.5dB and 3.7dB from capacity, respectively.

VIII. Conclusion

Low density lattice codes (LDLC) were introduced. LDLC are novel lattice codes that can approach capacity and be decoded efficiently. Good error performance within ~ 0.5dB from capacity at block length of 100,000 symbols was demonstrated, and convergence analysis was outlined. 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.

Acknowledgment

Support and interesting discussions with Ehud Weinstein are greatfully acknowledged.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423 and pp. 623-656, July and Oct. 1948.
[2] P. Elias, "Coding for noisy channels," in *IRE Conv. Rec.*, Mar. 1955, vol. 3, pt. 4, pp. 37-46.
[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*. 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," *Proc. IEEE Int. Conf. Communications*, pp. 1064–1070, 1993.
[7] R. de Buda, "The upper error bound of a new near-optimal code," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 441-445, July 1975.
[8] R. de Buda, "Some optimal codes have structure," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 893-899, Aug. 1989.
[9] T. Linder, Ch. Schlegel, and K. Zeger, "Corrected proof of de Buda's theorem," *IEEE Trans. Inform. Theory*, pp. 1735-1737, Sept. 1993.
[10] H. A. Loeliger, "Averaging bounds for lattices and linear codes," *IEEE Trans. Inform. 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. Inform. Theory*, pp. 273-278, Jan. 1998.
[12] U. Erez and R. Zamir, "Achieving 1/2 log(1 + 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. Inform. Theory*, vol. IT-33, pp. 177-195, Mar. 1987.
[14] G. D. Forney, Jr., "Coset codes-Part I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, pp. 1123-1151, Sept. 1988.
[15] O. Shalvi, N. Sommer and M. Feder, "Signal Codes," *proceedings of the Information theory Workshop*, 2003, pp. 332–336.
[16] J. H. Conway and N. J. Sloane, *Sphere Packings, Lattices and Groups*. New York: Springer, 1988.
[17] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Trans. Inform. Theory*, vol. 40, pp. 409-417, Mar. 1994.
[18] N. Sommer, M. Feder and O. Shalvi, "Low Density Lattice Codes," in preparation.
[19] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematic (SIAM), 2nd edition, 2003.