Received 23 July 2023; accepted 13 August 2023. Date of publication 15 August 2023; date of current version 1 September 2023.

Digital Object Identifier 10.1109/OJCOMS.2023.3305480

Efficient Decoder Design for Low-Density Lattice Codes From the Lattice Viewpoint

XUEBO WANG (Student Member, IEEE), AND WAI HO MOW (Senior Member, IEEE)

Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

CORRESPONDING AUTHOR: X. WANG (e-mail: xwangdb@connect.ust.hk)

This work was supported by the Hong Kong Research Grants Council under Project GRF 16214422.

Part of this work was presented at the 2019 IEEE Global Communication Conference [1] [DOI: 10.1109/GLOBECOM38437.2019.9013874].

ABSTRACT Low-density lattice codes (LDLCs) achieve near-capacity performance on additive white Gaussian noise (AWGN) channels. The M-Gaussian decoder is the state-of-the-art message passing decoder for LDLCs in terms of the error performance. However, this decoder has complexity $O(M^{d-1})$ with messages represented by Gaussian mixtures, where d is the degree of an LDLC and M is the number of Gaussian functions for approximating each check node message. In this paper, we establish the correspondence between Gaussian functions for approximating a variable node message and points of a certain lattice. Based on this lattice viewpoint, the problem of approximating a variable node message is formulated as a lattice point enumeration (LPE) problem. Then, an LPE decoder with linear complexity $O(d)$ is proposed. Our simulation results validate that the LPE decoder achieves almost the same error performance as the M-Gaussian decoder.

INDEX TERMS Low-density lattice codes, message passing, decoding complexity, lattice point enumeration.

I. INTRODUCTION

ACCORDING to Shannon's channel coding theorem, a code can be decoded at an arbitrarily small error probability if the transmission rate is lower than the channel capacity [2]. However, Shannon's theorem was proved by using a random code. Later, various structured codes were proposed, among which lattice codes possess great potential for practical applications and theoretical generalization [3].

Low-density lattice codes (LDLCs) were proposed in [4] and were shown to achieve near-capacity performance on AWGN channels. Several studies on relay networks using LDLCs were presented in [5], [6], [7], [8], [9], [10], [11]. Analogous to the well-known low-density parity check (LDPC) codes [12], LDLCs can be decoded using a message passing decoder. However, the messages exchanged between check nodes and variable nodes are probability density functions (PDFs), instead of log likelihood ratios.

Several LDLC message passing decoders [4], [13], [14], [15], [16], [17] have been proposed in the past two decades. The decoder in [4] exhibits the best-known error performance. Each PDF message in [4] is approximated by a vector of sampled values of the PDF. However, to provide a sufficient approximation accuracy, the required vector length for each message is impractically large, e.g., 1024. Later, Gaussian mixtures are used for approximating the messages. To keep the complexity affordable, the Gaussian mixture reduction (GMR) decoder was proposed in [13]. Specifically, multiple Gaussian functions of a mixture are approximated by a single Gaussian function. By applying the GMR, each message at most contains T Gaussian functions. The GMR decoder in [13] has complexity $O(n \cdot t \cdot d \cdot K^2 \cdot T^4)$, where n is the code length, t is the number of iterations, d is the degree of the LDLC and K is called the number of replications for check node messages. The GMR decoder was further improved in [14] with complexity $O(n \cdot t \cdot d \cdot K \cdot T^3)$. Nevertheless, the GMR is applied to every intermediate message and the GMR decoders [13], [14] may result in undesirably complicated implementation. Besides, manually adjusting two parameters is always needed for the GMR decoder to achieve the best performance. It is reported in [14] that the GMR decoder has a performance loss of 0.1 – 0.2 dB compared to the best-known error performance.

TABLE 1. Comparison of different Gaussian-mixture-based decoders for the LDLC of code length $n = 10^4$ and degree $d = 7$.

Decoder	Complexity	Gap from the capacity[1](dB)
GMR [13]	$O(n \cdot t \cdot d \cdot K^2 \cdot T^4)$	0.9
GMR [14]	$O(n \cdot t \cdot d \cdot K \cdot T^3)$	0.9
M-Gaussian [15]	$O(n \cdot t \cdot M^{d-1})$	0.8
Decoder in [17]	$O(n \cdot t \cdot d)$	1.3
LPE (proposed)	$O(n \cdot t \cdot d)$	0.8

[1] The gap from capacity is evaluated at the SER of 10^{-5}. The result in Fig. 9 in [15] is used for the GMR and M-Gaussian decoders. For the decoder in [17], the value is estimated by adding 0.3 dB to that of the 2-Gaussian decoder.

Compared to the GMR decoders, the M-Gaussian decoder proposed in [15] only requires the selection of one integer parameter M. Each check node message is approximated with a mixture containing M Gaussian functions. Computing each variable node message involves a product of $d-1$ check node messages. As a result, the complexity of the M-Gaussian decoder is $O(n \cdot t \cdot M^{d-1})$. The best-known error performance is achieved by the 2-Gaussian decoder for low-to-moderate LDLC dimensions, while the 3-Gaussian decoder is required for moderate-to-high dimensions (see [15, Fig. 9]). Later, a shuffled version of the M-Gaussian decoder was proposed in [16] with average complexity $O(n \cdot t \cdot 1.4^{d-1})$. Compared to *parallel* message passing decoders which update variable node messages in parallel, *shuffled* message passing decoders update variable node messages in a specified sequence. As a consequence, the messages updated later benefit from the information of the messages updated earlier. Thus, the shuffled decoders enjoy faster decoding convergence than the parallel counterparts. It is very likely to explore the shuffled design for many message passing decoders. However, in this paper, we will focus on the design of a parallel message passing decoder for LDLCs. The shuffled version is out of the scope of this paper. To further reduce decoding complexity, a faster decoder was proposed in [17] with complexity $O(n \cdot t \cdot d)$. Each variable node message is approximated with a mixture containing at most two Gaussian functions. However, compared to the 2-Gaussian decoder, the decoder in [17] has performance loss of 0.2 dB and 0.3 dB in the waterfall region for $n = 10^3$ and $n = 10^4$, respectively (see [17, Fig. 3]). The complexity and error performance of Gaussian-mixture-based decoders are summarized in Table 1. The error performance is evaluated by the distance from the channel capacity at the symbol error rate of 10^{-5}.

There exists a clear trade-off between the error performance and the complexity of existing decoders [15], [17]. Approximating each variable node message with more Gaussian functions leads to better error performance but higher complexity. The opposite holds when approximating the variable node message with fewer Gaussian functions. However, the M-Gaussian decoder may consider many unimportant Gaussian functions for approximating messages. Then, an intriguing question is, *is it possible that a decoder with polynomial (or even linear) complexity in d can achieve the best-known performance as the M-Gaussian decoder?* The difficulty in answering this question lies in the lack of a flexible scheme for choosing Gaussian functions for approximating variable node messages. Our work in this paper provides an affirmative answer. We first mathematically establish the correspondence between the Gaussian functions used to approximate a variable node message and the points of a certain lattice. This lattice viewpoint converts the problem for approximating messages to a lattice point enumeration (LPE) problem which can be solved by the list sphere decoding (LSD) algorithm [18], [19]. As a result, the number of Gaussian functions for approximating messages can be flexibly chosen by adjusting the radius of the sphere. The lattice viewpoint serves as the foundation for our decoder design. Contributions of this work are summarized as follows:

- We formulate the problem of approximating a variable node message as an LPE problem. Whether a Gaussian function is essential for approximating the variable node message depends on the distance between its corresponding lattice point and a query point. From this lattice viewpoint, we establish a unified framework for interpreting some existing decoders [15], [17]. By geometrically interpreting the Gaussian functions used for approximating a variable node message in various decoders, it can be deduced that the error performance of the M-Gaussian decoder for any value of M can be approached by a decoder that enumerates the lattice points within a sufficiently large sphere.
- The lattice viewpoint makes it possible to allow only essential Gaussian functions to be considered for approximating a variable node message. For efficiently addressing the LPE problem, a simplified LSD algorithm is derived by exploiting the relationship between diagonal and off-diagonal entries of the generator matrix of the underlying lattice. Based on the LSD algorithm, an LPE decoder is proposed for decoding LDLCs.
- The complexity of the LPE decoder is dominated by that of the LSD algorithm. With a judiciously chosen search sphere, we show that the LPE decoder achieves the complexity $O(n \cdot t \cdot d)$ and almost no performance loss compared to the M-Gaussian decoder. In addition, a complexity comparison of the LPE decoder and the M-Gaussian decoder in terms of the required number of floating-point operations is provided to demonstrate the superior efficiency of the LPE decoder.

Part of this work was presented in a conference version [1]. In this paper, different from [1], a rigorous derivation of the lattice viewpoint for the decoder design is presented. Besides, geometrical interpretation of different decoders is demonstrated. Further, performance analysis and numerical evaluation for the proposed LPE decoder are provided.

The rest of this paper is organized as follows. Section II introduces the preliminaries of LDLCs and basic operations on Gaussian and Gaussian mixtures as well as the LSD algorithm. In Section III, we present a detailed derivation of the lattice viewpoint for LDLC decoding, and geometrical interpretation of different decoders. A simplified LSD algorithm and the LPE decoder are presented in Section IV. The analysis on the complexity and required number of floating-point operations of the LPE decoder is provided in Section V. Numerical results are shown in Section VI. Finally, Section VII concludes the paper.

Notation: Non-bold italic letters denote scalars. Boldface lowercase and uppercase letters denote vectors and matrices, respectively. The function $\mathrm{sgn}(a)$ is 1 if the scalar a is positive and -1 if it is negative. The Hadamard product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is represented by $\mathbf{a} \odot \mathbf{b}$. The Hadamard inverse of the vector $\mathbf{a}$ is denoted by $\mathbf{a}^{\circ -1}$ [20]. Let $\mathbf{A}^T$ and $\mathbf{A}^{-1}$ denote the transpose and inverse of the matrix $\mathbf{A}$, respectively. Let $\mathrm{diag}(\mathbf{a})$ denote the square diagonal matrix with the elements of vector $\mathbf{a}$ on the main diagonal. We use $\mathcal{N}(x; m, v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(x-m)^2}{2v}}$ to denote the Gaussian function of the random variable x with the mean m and the variance v. The convolution of two functions $f(x)$ and $g(x)$ is denoted by $f(x) * g(x)$.

II. PRELIMINARIES

This section presents necessary background knowledge.

A. LATTICES AND LATTICE CODES

An n-dimensional lattice Λ, defined by an $n \times n$ full-rank generator matrix $\mathbf{G}$, is a discrete additive subgroup of $\mathbb{R}^n$. A lattice point $\mathbf{x} \in \Lambda$ is an integral linear combination of all columns of $\mathbf{G}$

$$\mathbf{x} = \mathbf{G}\mathbf{b}, \tag{1}$$

where $\mathbf{b} \in \mathbb{Z}^n$ is the information column vector.

If a lattice point is transmitted as a codeword over the AWGN channel, the channel observation is

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{2}$$

where $\mathbf{n}$ is the additive Gaussian noise random vector with zero mean and covariance matrix $\sigma^2\mathbf{I}$. For simplicity, in this paper, we consider the power-unconstrained AWGN channel. Thus, any lattice point could be a codeword. Since the definition of the signal-to-noise ratio (SNR) becomes meaningless without power constraint, the volume-to-noise ratio (VNR) [21], measured by the lattice constellation density and σ^2, is considered for the power-unconstrained channel:

$$\mathrm{VNR} = \frac{|\det(\mathbf{G})|^{2/n}}{2\pi e\sigma^2}. \tag{3}$$

The definition of VNR generalizes the concept of channel capacity by measuring the maximum lattice point density that can be recovered. That is, the unconstrained AWGN channel capacity is achieved if the error probability can be arbitrarily small when $\sigma^2 \leq \frac{|\det(\mathbf{G})|^{2/n}}{2\pi e}$.

B. LOW-DENSITY LATTICE CODES

A low-density lattice code (LDLC) [4] is characterized by the inverse of its generator matrix $\mathbf{H} = \mathbf{G}^{-1}$. The matrix $\mathbf{H}$ is sparse and is named the check matrix of the LDLC. More explicitly, every row and column of $\mathbf{H}$ has only d nonzero entries where $d \ll n$ is named the degree of the LDLC. All nonzero entries are assigned random signs. Besides, The absolute values of the d nonzero entries in each row or column are given by a generating sequence $\{1, w, \ldots, w\}$ of d elements, where $w = \sqrt{\frac{\alpha}{d-1}}$ with a constant α satisfying $0 < \alpha < 1$ [13], [14], [15], [16], [17].

1) LDLC MESSAGE PASSING DECODER

Since $\mathbf{H} = \mathbf{G}^{-1}$, the check equations for an LDLC are naturally represented by $\mathbf{H}\mathbf{x} = \mathbf{b}$. Based on $\mathbf{H}$, a Tanner graph is constructed with check nodes representing the check equations and variable nodes the coordinates of $\mathbf{x}$. An edge between the i-th check node and the j-th variable node of the Tanner graph exists if $H_{i,j} \neq 0$.

The LDLC message passing decoder derived in [4] is shown below. During one iteration, the following steps are performed:

- Initialization: the k-th variable node sends its connected check nodes the channel message

$$f_k(x) = \mathcal{N}\left(x; y_k, \sigma^2\right) \tag{4}$$

  for $k = 1, \ldots, n$.
- Check node messages: After receiving the messages from d connected variable nodes, the j-th check node calculates the message sent to its i-th connected variable node in three steps, for $j = 1, \ldots, n$ and $i = 1, \ldots, d$.
  1) Convolution step:

$$\begin{aligned}\tilde{p}_{j,j_i}(x) &= f_{j_1,j}\left(\frac{x}{H_{j,j_1}}\right) * \cdots * f_{j_{i-1},j}\left(\frac{x}{H_{j,j_{i-1}}}\right) \\ &\quad * f_{j_{i+1},j}\left(\frac{x}{H_{j,j_{i+1}}}\right) * \cdots * f_{j_d,j}\left(\frac{x}{H_{j,j_d}}\right)\end{aligned} \tag{5}$$

     where j_i is the index of the i-th variable node connected with the j-th check node, and $f_{j_i,j}(x)$ is the message sent from the j_i-th variable node to the j-th check node.
  2) Stretching step:

$$\hat{p}_{j,j_i}(x) = \tilde{p}_{j,j_i}\left(-H_{j,j_i}x\right). \tag{6}$$

  3) Periodic extension:

$$p_{j,j_i}(x) = \sum_{z\in\mathbb{Z}} \hat{p}_{j,j_i}\left(x + \frac{z}{H_{j,j_i}}\right). \tag{7}$$

  Then, the j-th check node will send the message $p_{j,j_i}(x)$ to its i-th connected variable node.

- Variable node messages: Once obtaining the messages from d connected check nodes, the k-th variable node calculates the message sent to its i-th connected check node in two steps, for $k = 1, \ldots, n$ and $i = 1, \ldots, d$.
  1) Product step:
$$\tilde{f}_{k,k_i}(x) = \mathcal{N}\left(x; y_k, \sigma^2\right) \prod_{l=1, l\neq i}^{d} p_{k_l,k}(x), \tag{8}$$
  where k_l is the index of the l-th check node connected with the k-th variable node.
  2) Normalization:
$$f_{k,k_i}(x) = \frac{\tilde{f}_{k,k_i}(x)}{\int_{-\infty}^{\infty} \tilde{f}_{k,k_i}(x)dx}. \tag{9}$$
- Final decision: when the maximum number of iterations is reached, the final step is performed for every variable node:
$$f_k^f(x) = \mathcal{N}\left(x; y_k, \sigma^2\right) \prod_{l=1}^{d} p_{k_l,k}(x), \tag{10}$$
  the codeword is estimated as $\hat{\mathbf{x}}$ with each coordinate
$$\hat{x}_k = \arg\max_x f_k^f(x), \tag{11}$$
  and the integer message vector is detected as
$$\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil. \tag{12}$$

For the derivation of the above decoder, interested readers are referred to [4].

2) CONVERGENCE OF THE VARIANCES OF VARIABLE NODE MESSAGES

As the iteration proceeds, the variable node messages tend to be single Gaussian functions. Let V_i denote the variance of the variable node message that is sent from the k-th variable node to its i-th connected check node. The convergence of V_i is [4]
$$\lim_{t\to\infty} V_i = \begin{cases} 0 & \text{if } |H_{k_i,k}| = w \\ \sigma^2(1-\alpha) & \text{if } |H_{k_i,k}| = 1 \end{cases}. \tag{13}$$

C. OPERATIONS ON GAUSSIAN FUNCTIONS AND GAUSSIAN MIXTURES

To calculate variable node messages, the product of two Gaussian functions and the moment matching approximation are described below.

1) PRODUCT OF TWO GAUSSIAN FUNCTIONS

Given $\mathcal{N}(x; m_1, v_1)$ and $\mathcal{N}(x; m_2, v_2)$, the product of them is a weighted Gaussian function $c\mathcal{N}(x; m, v)$ where
$$v = \left(\frac{1}{v_1} + \frac{1}{v_2}\right)^{-1}, \tag{14}$$
$$m = v\left(\frac{m_1}{v_1} + \frac{m_2}{v_2}\right), \tag{15}$$
$$c = \frac{1}{\sqrt{2\pi(v_1+v_2)}} e^{-\frac{(m_1-m_2)^2}{2(v_1+v_2)}}. \tag{16}$$

2) MOMENT MATCHING APPROXIMATION

A Gaussian mixture is a weighted sum of Gaussian functions, i.e., $f(x) = \sum_{i=1}^{N} c_i \mathcal{N}(x; m_i, v_i)$ with $\sum_{i=1}^{N} c_i = 1$. The moment matching approximation is commonly used in [15], [16], [17] to approximate a Gaussian mixture using a single Gaussian function by minimizing the Kullback-Leiber divergence between them [15]. The mean and variance of the approximating Gaussian function are
$$\bar{m} = \sum_{i=1}^{N} c_i m_i \tag{17}$$
and
$$\bar{V} = \sum_{i=1}^{N} c_i\left(m_i^2 + v_i\right) - \left(\sum_{i=1}^{N} c_i m_i\right)^2, \tag{18}$$
respectively.

D. LIST SPHERE DECODING ALGORITHM

Given an arbitrary point $\mathbf{q} \in \mathbb{R}^n$ and a lattice Λ with the generator matrix $\mathbf{G}$, the LPE problem is to find a list of lattice points satisfying
$$\|\mathbf{q} - \mathbf{G}\mathbf{z}\|^2 < \beta^2. \tag{19}$$

The list sphere decoding (LSD) algorithm, the variant of SD [18], [19] with a list output, is an efficient tree search for solving the LPE problem. Suppose that $\mathbf{G}$ is an upper triangular matrix, the search can be performed layer by layer using successive interference cancellation. The Euclidean distance between the query point $\mathbf{q}$ and the current searched point is used as the metric. Let D_k^2 denote the squared partial Euclidean distance at the k-th layer. The metric is accumulated as
$$D_k^2 = D_{k+1}^2 + \left(q_k - \sum_{i=k}^{n} G_{k,i} z_i\right)^2, \tag{20}$$
where $k = n, n-1, \ldots, 1$ and $D_{n+1} = 0$. In this paper, we consider the depth-first SD. The search starts from the n-th layer (root layer) to the first layer (leaf layer). If $D_k \geq \beta$ for $k = n-1, \ldots, 1$, we will move up to the $(k+1)$-th layer and update z_{k+1} following the Schnorr-Euchner ordering [22]. If $D_k < \beta$ for $k = n, \ldots, 2$, we will move down to the $(k-1)$-th layer for the search. If the algorithm reaches the first layer (leaf layer) and $D_1 < \beta$, the current searched lattice point is stored and the algorithm continues to seek another point satisfying (19). The algorithm will stop when $D_n \geq \beta$ and output a list of integer vectors $\mathbf{z}$.

In case $\mathbf{G}$ is not upper triangular, one could apply the QR decomposition to $\mathbf{G}$ such that $\mathbf{G} = \mathbf{U}\mathbf{R}$, where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. Then, the constraint (19) becomes $||\mathbf{U}^T\mathbf{q} - \mathbf{R}\mathbf{z}||^2 < \beta^2$.

III. LATTICE VIEWPOINT FOR LDLC DECODING

In this section, we first further derive the expression of variable node messages. Based on the derivation result, we then elaborate on the lattice viewpoint for LDLC decoding. Similar to what is done in [15], [23], we also apply the moment matching approximation to the variable node messages so that the messages exchanged between nodes are single Gaussian functions.

A. VARIABLE NODE MESSAGES WITH THE MOMENT MATCHING APPROXIMATION

As shown in (8), each variable node message is obtained by calculating the product of $d-1$ check node messages and the channel message. Without loss of generality, we focus on the message sent from the k-th variable node to the d-th connected check node:

$$\begin{aligned}\tilde{f}_{k,k_d}(x) &= \mathcal{N}\left(x; y_k, \sigma^2\right) \prod_{l=1}^{d-1} p_{k_l,k}(x) \\ &= \mathcal{N}\left(x; y_k, \sigma^2\right) \prod_{l=1}^{d-1} \sum_{z_l \in \mathbb{Z}} \mathcal{N}\left(x; a_l + \frac{z_l}{H_{k_l,k}}, v_l\right), \quad (21)\end{aligned}$$

where a_l and v_l are the mean and variance of $\hat{p}_{k_l,k}(x)$, the check node message before the periodic extension. For ease of presentation, in the sequel, we change the notations of some variables in (21) by letting $p_l(x) = p_{k_l,k}(x)$ and $h_l = H_{k_l,k}$. Then,

$$\tilde{f}_{k,k_d}(x) = \mathcal{N}\left(x; y_k, \sigma^2\right) \prod_{l=1}^{d-1} \sum_{z_l \in \mathbb{Z}} \mathcal{N}\left(x; a_l + \frac{z_l}{h_l}, v_l\right). \quad (22)$$

Due to the product step, $\tilde{f}_{k,k_d}(x)$ is a Gaussian mixture containing infinite Gaussian functions. For practical implementation, only a finite number of Gaussian functions can be considered. Assume that $\tilde{f}_{k,k_d}(x)$ is approximated by considering N Gaussian functions. Define N integer vectors $\mathbf{z}_s \in \mathbb{Z}^d$, for $s = 1, \ldots, N$, where $\mathbf{z}_s \triangleq [z_{s,1}, \ldots, z_{s,d}]^T$. The s-th Gaussian function is expressed as

$$c_s \mathcal{N}(x; m_s, V) = \prod_{l=1}^{d} \mathcal{N}\left(x; a_l + \frac{z_{s,l}}{h_l}, v_l\right), \quad (23)$$

where $\mathcal{N}(x; y_k, \sigma^2)$ is viewed as the d-th factor with $v_d \triangleq \sigma^2$, $a_d \triangleq y_k$, $z_{s,d} = 0$, $h_d = 1$. According to (14)–(16),

$$\frac{1}{V} = \sum_{l=1}^{d} \frac{1}{v_l}, \quad (24)$$

$$m_s = V\left(\sum_{l=1}^{d} \frac{a_l + z_{s,l}/h_l}{v_l}\right), \quad (25)$$

$$c_s = C e^{-\frac{V}{2} \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{(a_i + z_{s,i}/h_i - a_j - z_{s,j}/h_j)^2}{v_i v_j}}, \quad (26)$$

where C is a normalization factor such that $\sum_{s=1}^{N} c_s = 1$.

The mean (25) and scaling coefficient (26) can be written in a vectorized form as

$$m_s = V(\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T (\mathbf{h} \odot \mathbf{v})^{\circ -1} \quad (27)$$

and

$$c_s = C e^{-\frac{1}{2}(\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})}, \quad (28)$$

respectively, where $\mathbf{h} = [h_1, \ldots, h_{d-1}, 1]^T$, $\mathbf{a} = [a_1, \ldots, a_{d-1}, y_k]^T$, $\mathbf{v} = [v_1, \ldots, v_{d-1}, \sigma^2]^T$ and $\mathbf{Q} = \mathbf{Q}_1 - \mathbf{Q}_2$ with

$$\mathbf{Q}_1 \triangleq \operatorname{diag}\left(\left[\frac{1}{h_1^2 v_1}, \ldots, \frac{1}{h_{d-1}^2 v_{d-1}}, \frac{1}{\sigma^2}\right]\right) \quad (29)$$

and

$$\mathbf{Q}_2 \triangleq \begin{bmatrix} \frac{\sqrt{V}}{v_1 h_1} \\ \vdots \\ \frac{\sqrt{V}}{v_{d-1} h_{d-1}} \\ \frac{\sqrt{V}}{\sigma^2} \end{bmatrix} \left[\frac{\sqrt{V}}{v_1 h_1}, \ldots, \frac{\sqrt{V}}{v_{d-1} h_{d-1}}, \frac{\sqrt{V}}{\sigma^2}\right]. \quad (30)$$

The derivation of (28) from (26) is given in the Appendix. We can see that the s-th Gaussian function for approximating the variable node message corresponds to an integer vector $\mathbf{z}_s$.

Next, to simplify the message passed between nodes [15], [23], the moment matching approximation is used such that the variable node message is further approximated by a single Gaussian function. According to (17) and (18), for achieving good approximation by moment matching, we are keen on finding a list of $\mathbf{z}_s$ giving sufficiently large $c_s|m_s|$ and $c_s m_s^2$, i.e.,

$$\{\mathbf{z}_s : c_s|m_s| > \rho_1\} \cup \left\{\mathbf{z}_s : c_s m_s^2 > \rho_2\right\}, \quad (31)$$

where ρ_1 and ρ_2 are two scalar parameters and should be judiciously chosen to achieve a desired trade-off between the approximation accuracy and search complexity. According to (27) and (30), it is intriguing to note that

$$m_s^2 / V = (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_2 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}). \quad (32)$$

Then, we have

$$\begin{aligned} &c_s|m_s| \\ &= C e^{-\frac{1}{2}(\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})} \sqrt{V (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_2 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})} \end{aligned} \quad (33)$$

and

$$\begin{aligned} &c_s m_s^2 \\ &= C V e^{-\frac{1}{2}(\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_2 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}). \end{aligned} \quad (34)$$

B. LATTICE VIEWPOINT

The straightforward search for (31) is infeasible due to the complicated expressions of $c_s|m_s|$ and $c_s m_s^2$. For the tractability of search, we alternatively consider searching the list of $\mathbf{z}_s$ defined by

$$\left\{\mathbf{z}_s : (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}) < \beta^2\right\}. \tag{35}$$

The search for (35) should find a list of $\mathbf{z}_s$ corresponding to Gaussian functions having sufficiently large $c_s > Ce^{-\frac{1}{2}\beta^2}$. Nevertheless, in what follows, we will show that the list of $\mathbf{z}_s$ defined by (31) can still be found by the search for (35). Based on (32), we have

$$m_s^2/V > (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_1 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}) - \beta^2. \tag{36}$$

Thus, the searched integer vectors in (35) correspond to Gaussian functions with

$$c_s m_s^2 > CV\Big((\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_1 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}) - \beta^2\Big) e^{-\frac{1}{2}\beta^2} \tag{37}$$

and

$$c_s|m_s| > C\sqrt{V\big((\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_1 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}) - \beta^2\big)} e^{-\frac{1}{2}\beta^2}. \tag{38}$$

Note that the search region associated with (35) is a sphere of radius β. Then, the list of $\mathbf{z}_s$ defined by (31) can be a subset of (35) as long as the sphere of radius β covers the search region associated with (31).

By applying the Cholesky factorization to $\mathbf{Q}$, the constraint in (35) becomes

$$\|\mathbf{R}(\mathbf{z} + \mathbf{h} \odot \mathbf{a})\|^2 < \beta^2, \tag{39}$$

where the subscript s of $\mathbf{z}$ is omitted for ease of presentation and $\mathbf{R}$ is an upper triangular matrix satisfying $\mathbf{Q} = \mathbf{R}^T\mathbf{R}$. Note that each lattice point $\mathbf{Rz}$ corresponds to a Gaussian function for approximating a variable node message. We now have a lattice viewpoint to interpret the problem of finding essential Gaussian functions for approximating the variable node messages. Clearly, searching a list of $\mathbf{z}$ with the constraint (39) is an LPE problem with the query point $\mathbf{q} = -\mathbf{R}(\mathbf{h} \odot \mathbf{a})$. Then, the LSD algorithm can be utilized to enumerate the desired integer vectors.

C. GEOMETRICAL INTERPRETATION OF DIFFERENT DECODERS

From the lattice viewpoint, we can geometrically compare the number of Gaussian functions used for approximating each variable node message in the M-Gaussian decoder [15] and the low complexity decoder [17]. Each Gaussian function corresponds to a lattice point. A 2-dimensional example is depicted in Fig. 1 where calculating each variable node message needs the product of two check node messages ($d = 3$). The solid circle dots represent lattice points $\mathbf{Rz}$ and the red star is the query point $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$.

For the M-Gaussian decoder, each check node message is a mixture containing M Gaussian functions. When $M = 2$, four lattice points on the blue solid edges, which are determined by considering the two closest lattice points to the query point in each lattice dimension, are used for approximating each variable node message. Similarly, nine lattice points within the parallelogram shaped by the red dashed edges are considered by the 3-Gaussian decoder.

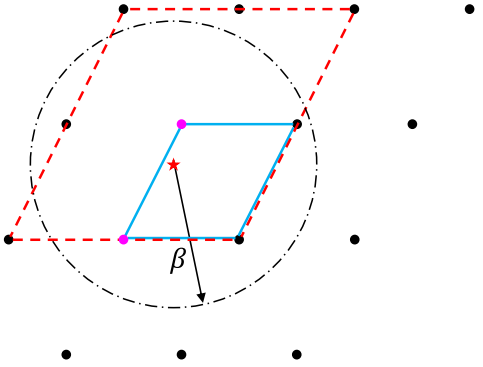


FIGURE 1. Geometrical interpretation of the M-Gaussian decoder [15] and the low complexity decoder [17] from the lattice viewpoint: solid circle dots represent the lattice points Rz and the red star is for the query point −R(h ⊙ a). The 2-Gaussian decoder considers four lattice points on the edges of the parallelogram enclosed by blue solid edges. The 3-Gaussian decoder considers nine lattice points within and on the edges of the parallelogram shaped by red dashed edges. The decoder in [17] considers two lattice points of purple color. The LSD algorithm enumerates five lattice points within the dash-dot circle of the radius β.

The decoder in [17] focuses on the parallelogram shaped by the blue solid edges, i.e., the region considered by the 2-Gaussian decoder. However, at most two lattice points (marked by purple color in Fig. 1) within this region are considered by the decoder in [17]. The two lattice points are approximately to be the nearest to the query points. Clearly, this decoder sacrifices the approximation accuracy for the variable node messages.

A trade-off between decoding performance and complexity exists in different decoders. The M-Gaussian decoder greedily considers all lattice points within a parallelogram, while the decoder in [17] only considers a fixed number of them. Consequently, the M-Gaussian decoder has better error performance but higher complexity than the latter one. Besides, for approximating variable node messages, unimportant Gaussian functions may be used if M is too large and essential Gaussian functions may be ignored for small M. For achieving a flexible trade-off, in this paper, we consider the lattice points within a sphere centering at the query point, which can be searched by applying the LSD algorithm. In the example of Fig. 1, five lattice points within the sphere of radius β are found by the LSD algorithm, which considers one more essential Gaussian function than the 2-Gaussian decoder and discards four unimportant Gaussian functions used in the 3-Gaussian decoder.

The interpretation from the lattice viewpoint indicates that the performance of the M-Gaussian decoder can be approached by choosing lattice points within a sufficiently large sphere.

IV. SIMPLIFIED LSD ALGORITHM AND THE PROPOSED LPE DECODER

In this section, based on the specific relationship between diagonal and off-diagonal entries of $\mathbf{R}$, a simplified LSD algorithm is derived first. Then, the LPE decoder is introduced.

A. GENERATOR MATRIX OF THE LATTICE FOR THE LPE PROBLEM

This subsection provides an explicit expression of $\mathbf{R}$ which is the generator matrix of the lattice involved in the LPE problem presented in Section III-B. Thanks to the specific structure of $\mathbf{Q}$ (see (29) and (30)), we first have

$$\mathbf{S}^{-1}\mathbf{Q}\mathbf{S}^{-1} = \mathbf{I} - \mathbf{t}\mathbf{t}^T = \tilde{\mathbf{R}}^T\tilde{\mathbf{R}}, \tag{40}$$

where $\mathbf{t} \triangleq [\mathrm{sgn}(h_1)\sqrt{V/v_1}, \ldots, \mathrm{sgn}(h_d)\sqrt{V/v_d}]^T$ and $\mathbf{S} \triangleq \mathrm{diag}([(h_1^2 v_1)^{-1/2}, \ldots, (h_d^2 v_d)^{-1/2}])$. The matrix $\tilde{\mathbf{R}}$ is the Cholesky factor of $\mathbf{I} - \mathbf{t}\mathbf{t}^T$, which can be efficiently solved by the method proposed in [24]. For obtaining $\mathbf{R}$, one more step is needed:

$$\mathbf{R} = \tilde{\mathbf{R}}\mathbf{S}. \tag{41}$$

Then, the entries of $\mathbf{R}$ are

$$R_{ij} = \begin{cases} \frac{1}{\sqrt{h_j^2 v_j}} \sqrt{\frac{1-\sum_{l=1}^{i} t_l^2}{1-\sum_{l=1}^{i-1} t_l^2}} & j = i \\ \frac{1}{\sqrt{h_j^2 v_j}} \frac{-t_i t_j}{\sqrt{1-\sum_{l=1}^{i} t_l^2}\sqrt{1-\sum_{l=1}^{i-1} t_l^2}} & i < j \le d \end{cases} \tag{42}$$

where $1 \le i \le d$ and $\sum_{l=1}^{0} = 0$.

It is notable that $R_{dd} = 0$ since $\sum_{l=1}^{d} t_l^2 = 1$, which means that the last row of $\mathbf{R}$ is all zeros. However, since z_d is fixed to be zero according to our formulation in (23), the first $d-1$ rows of $\mathbf{R}$ are sufficient for applying the LSD algorithm.

B. SEARCH RADIUS FOR THE LSD ALGORITHM

The choice of search radius will affect the complexity of the LSD algorithm since it affects the number of visited nodes inside the search space. Note that the basis length of the lattice spanned by $\mathbf{R}$ is

$$\left\|\mathbf{R}_{:,j}\right\| = \sqrt{\frac{1}{h_j^2 v_j}\left(1 - t_j^2\right)} < \sqrt{\frac{1}{h_j^2 v_j}}. \tag{43}$$

We could heuristically set the initial choice of the search radius as the upper bound of the largest basis length, $\max_j \sqrt{1/(h_j^2 v_j)}$. However, according to (13), v_j will converge to zero if $|h_j| = 1$, which causes an infinitely large search radius. Thus, we set the initial choice of the search radius by only considering $|h_j| = w$, i.e.,

$$\beta_1 = \max_j \sqrt{\frac{1}{h_j^2 v_j}} \quad \text{s.t.} \quad |h_j| = w. \tag{44}$$

By considering the right-hand side of (28) as a function of $\mathbf{z}_s$, we notice that (28) indicates a lattice Gaussian distribution [25] of variance 1 centered at $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$. Then, the scaling coefficient c_s equals the probability of the lattice point $\mathbf{R}\mathbf{z}_s$. The nearest lattice point to $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$ should have the largest probability. From this perspective, we should avoid searching for any lattice points with relatively small probabilities compared to the nearest lattice point to $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$. However, since it is usually difficult to know the nearest lattice point to the query point before the search, we consider the Babai point [26] instead. Let D_B denote the distance between the Babai point and the query point. The probability of the Babai point is $Ce^{-\frac{1}{2}D_B^2}$. We set a threshold ϵ such that only lattice points with probability larger than $\epsilon Ce^{-\frac{1}{2}D_B^2}$ are searched with $0 < \epsilon < 1$. Equivalently, the corresponding search radius β_2 should satisfy $Ce^{-\frac{1}{2}\beta_2^2} = \epsilon Ce^{-\frac{1}{2}D_B^2}$. Then, $\beta_2 = \sqrt{D_B^2 + 2\ln(1/\epsilon)}$. By taking (44) into consideration, we set the search radius

$$\beta = \min\left\{\beta_1, \sqrt{D_B^2 + 2\ln(1/\epsilon)}\right\}. \tag{45}$$

In this paper, we set $\epsilon = 10^{-5}$. Since we choose to apply the LSD algorithm with the Schnorr-Euchner ordering, the Babai point must be the first candidate to be enumerated. Thus, finding out D_B introduces no additional complexity.

By simulation, at the first iteration, we observe that the LSD algorithm may output an empty list of candidates with the search radius (44) at a slight probability. For this rare case, the affected variable node message just keeps unchanged and waits for being updated at the next iteration. The probability of obtaining an empty list will be presented in Section VI-C.

C. SIMPLIFIED LSD ALGORITHM FOR LDLC DECODING

The LSD algorithm can be simplified by exploiting the specific expression of R_{ij} and R_{ii} in (42). Denoting $\mathbf{p} = \mathbf{h} \odot \mathbf{a}$, the left-hand side of (39) can be further expanded as

$$\begin{aligned} \|\mathbf{R}(\mathbf{z} + \mathbf{p})\|^2 &= \sum_{i=1}^{d-1} R_{ii}^2 \left(z_i + p_i + \frac{1}{R_{ii}} \sum_{j=i+1}^{d} R_{ij}(z_j + p_j) \right)^2 \\ &= \sum_{i=1}^{d-1} R_{ii}^2 (z_i - \gamma_i)^2, \end{aligned} \tag{46}$$

where

$$\gamma_i = -p_i - \frac{1}{R_{ii}} \sum_{j=i+1}^{d} R_{ij}(z_j + p_j) \tag{47}$$

for $i = 1, \ldots, d-1$. For the LSD algorithm, we mainly focus on simplifying the calculation of γ_i in this subsection. Define two d-dimensional vectors $\mathbf{f}$ and $\mathbf{g}$ with elements

$$f_i = 1 - \sum_{l=1}^{i} t_l^2 \tag{48}$$

and

$$g_i = \sqrt{h_i^2 v_i}, \tag{49}$$

for $i = 1, \ldots, d$. According to (42),

$$\frac{R_{ij}}{R_{ii}} = -\frac{\sqrt{h_i^2 v_i}}{\sqrt{h_j^2 v_j}} \cdot \frac{t_i t_j}{1 - \sum_{l=1}^{i} t_l^2} = -\frac{g_i t_i t_j}{f_i g_j}. \tag{50}$$

By substituting (50) into (47), we have

$$\gamma_i = -p_i + \frac{g_i t_i}{f_i} \sum_{j=i+1}^{d} \frac{t_j}{g_j}(z_j + p_j). \tag{51}$$

To further simplify the calculation of γ_i, we introduce a d-dimensional vector $\mathbf{u}$ with $u_d = \sqrt{V} a_d / v_d$. For $i = 1, \ldots, d-1$, once z_i is found, u_i is computed in a recursive manner:

$$u_i = \frac{t_i}{g_i}(z_i + p_i) + u_{i+1}. \tag{52}$$

Substituting (52) back into (51),

$$\gamma_i = -p_i + \frac{g_i t_i}{f_i} u_{i+1}, \tag{53}$$

for $i = 1, \ldots, d-1$.

By introducing the vectors $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{u}$, we simplify the calculation of γ_i. By (42), (48), and (49), we further have $R_{ii}^2 = \frac{f_i}{g_i^2 f_{i-1}}$. Now, the inputs to the LSD algorithm include $\mathbf{f}$, $\mathbf{g}$, $\mathbf{p}$, $\mathbf{t}$, β, and R_{ii}^2 for $i = 1, \ldots, d-1$. Note that the complexities for computing all these inputs are only $O(d)$. Besides, the complexities for computing each u_i and γ_i are $O(1)$ due to (52). For ease of direct implementation, the simplified LSD algorithm is summarized in Algorithm 1. Assume that N lattice points $\mathbf{R}\mathbf{z}_i$ are searched by the LSD algorithm, for $i = 1, \ldots, n$. Two lists will be output by the algorithm. The first list $\mathcal{L}$ subsumes the N integer vectors $\mathbf{z}_i$ for $i = 1, \ldots, n$, corresponding to the N searched lattice points. Mathematically, $\mathcal{L} \triangleq \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. The second list $\mathcal{D}$ contains N squared Euclidean distances between lattice points $\mathbf{R}\mathbf{z}_i$ and the point $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$, i.e., $\mathcal{D} \triangleq \{||\mathbf{R}(\mathbf{z}_1 + \mathbf{h} \odot \mathbf{a})||^2, \ldots, ||\mathbf{R}(\mathbf{z}_N + \mathbf{h} \odot \mathbf{a})||^2\}$.

Algorithm 1: Simplified List Sphere Decoding

Input: $\mathbf{f}$, $\mathbf{g}$, $\mathbf{p}$, $\mathbf{t}$, u_d, R_{ii}^2, β (see (45), (48), (49) and (52))
Output: $\mathcal{L}$, $\mathcal{D}$

$\mathcal{L} = \emptyset, \mathcal{D} = \emptyset, k = d-1, dist_d = 0$
$\gamma_k = -p_k + t_k g_k u_{k+1}/f_k$ // see (53)
$z_k = \lfloor \gamma_k \rceil$
$s_k = \mathrm{sgn}(\gamma_k - z_k)$
$dist_k = dist_{k+1} + R_{kk}^2 (z_k - \gamma_k)^2$ // update the search metric
while $k \leq d-1$ **do**
  if $dist_k < \beta^2$ **then**
    if $k == 1$ **then**
      add $\mathbf{z}$ into $\mathcal{L}$ and $dist_1$ into $\mathcal{D}$
      $z_k = z_k + s_k$
      $s_k = -s_k - \mathrm{sgn}(s_k)$
      $dist_k = dist_{k+1} + R_{kk}^2 (z_k - \gamma_k)^2$ // update the search metric
    else
      $u_k = t_k(z_k + p_k)/g_k + u_{k+1}$ // see (52)
      $k = k - 1$ // move down to the next layer
      repeat steps 2-5
  else
    if $k == d-1$ **then**
      return
    else
      $k = k + 1$ // move up to the previous layer
      repeat steps 10-12

D. LPE DECODER

The proposed decoder based on the lattice viewpoint is introduced in this subsection. For computing check node messages, the convolution and stretching steps are the same as (5) and (6), respectively. Since the integers in the periodic extension step (7) are found by the LSD algorithm under our formulation, we can avoid the periodic extension step. Thus, only the steps for obtaining variable node messages are presented.

- Variable node messages: for computing the message sent from the k-th variable node to one of its connected check node,
  1) obtain the input messages sent from $d-1$ remaining connected check nodes:
  $$\hat{p}_l(x) = \mathcal{N}(x; a_l, v_l), \tag{54}$$
  where $l = 1, \ldots, d-1$.
  2) obtain the vectors $\mathbf{h} = [h_1, \ldots, h_{d-1}, 1]^T$, $\mathbf{a} = [a_1, \ldots, a_{d-1}, y_k]^T$ and $\mathbf{p} = \mathbf{h} \odot \mathbf{a}$.
  3) find the vectors $\mathbf{t}$, $\mathbf{f}$, $\mathbf{g}$ and the first $d-1$ squared diagonal elements of $\mathbf{R}$.
  4) find the list $\mathcal{L}$ of integer vectors $\mathbf{z}$ and the list $\mathcal{D}$ of squared distances between all obtained lattice points $\mathbf{R}\mathbf{z}$ and the query point $-\mathbf{R}(\mathbf{h} \odot \mathbf{a})$ by using Algorithm 1.
  5) If $\mathcal{L} = \emptyset$, the variable node message keeps unchanged and waits for being updated at the next iteration. Else, for each candidate in $\mathcal{L}$, calculate the corresponding Gaussian function according to (24), (25), and (28). Obtain the Gaussian mixture by taking the sum of these Gaussian functions. Note that the scaling coefficients in the mixture should be normalized by their sum. Then the variable node message is approximated by applying the moment matching approximation to the Gaussian mixture.

V. COMPLEXITY ANALYSIS FOR THE PROPOSED DECODER

A. COMPLEXITY ORDER ANALYSIS

The complexity order of our proposed decoder is determined by that of computing check node messages and variable node messages. As presented in Section IV-D, the complexity of computing a variable node message is further dominated by that of the LSD algorithm. In this subsection, we first analyze the complexity order of the simplified LSD algorithm. Then, the complexity of the proposed LPE decoder is presented.

1) COMPLEXITY ORDER OF SIMPLIFIED LSD ALGORITHM

We first analyze the complexity of the LSD algorithm which is the most complicated step for computing variable node messages. The LSD algorithm traverses $d-1$ layers and its complexity depends on the number of visited nodes at each layer. Each node at the k-th layer represents the integer z_k. For $k=1,\ldots,d-1$, we define

$$\mathcal{P}_k(\beta) \triangleq \left\{\mathbf{z}_{k:d-1}\in\mathbb{Z}^{d-k}:\left\|\mathbf{R}_{k:d-1,k:d-1}\mathbf{z}_{k:d-1}-\mathbf{q}_{k,d-1}\right\|<\beta\right\} \tag{55}$$

as the set of searched partial integer vectors $\mathbf{z}_{k:d-1}\triangleq[z_k,\ldots,z_{d-1}]^T$. Due to the constraint of the search radius β, visited nodes at the k-th layer consist of the nodes satisfying $\mathbf{z}_{k:d-1}\in\mathcal{P}_k(\beta)$ and those simultaneously satisfying $\mathbf{z}_{k:d-1}\notin\mathcal{P}_k(\beta)$ and $\mathbf{z}_{k+1:d-1}\in\mathcal{P}_{k+1}(\beta)$. Thus, the number of visited nodes at the k-th layer is $|\mathcal{P}_k(\beta)|+|\mathcal{P}_{k+1}(\beta)|$. The complexity for searching all nodes at the k-th layer is

$$(|\mathcal{P}_k(\beta)|+|\mathcal{P}_{k+1}(\beta)|)O(1). \tag{56}$$

Then, for all layers, the overall complexity is

$$\sum_{k=1}^{d-1}(|\mathcal{P}_k(\beta)|+|\mathcal{P}_{k+1}(\beta)|)O(1)=O(1)\sum_{k=1}^{d-1}|\mathcal{P}_k(\beta)|, \tag{57}$$

where we define $|\mathcal{P}_d(\beta)|=0$.

The cardinality $|\mathcal{P}_k(\beta)|$ can be estimated by calculating the volume ratio between the search sphere and the fundamental region of the lattice spanned by $\mathbf{R}_{k:d-1,k:d-1}$ [24], i.e.,

$$|\mathcal{P}_k(\beta)|\approx\frac{\beta^{d-k}}{\prod_{i=k}^{d-1}R_{ii}}\mathsf{V}_{d-k}, \tag{58}$$

where V_{d-k} is the volume of a $(d-k)$-dimensional ball of unit radius and is upper bounded by 5.2638 [24]. According to (42), we have

$$\prod_{i=k}^{d-1}R_{ii}=\sqrt{\frac{t_d^2}{1-\sum_{l=1}^{k-1}t_l^2}}\prod_{i=k}^{d-1}\frac{1}{\sqrt{h_i^2v_i}}. \tag{59}$$

Then,

$$\frac{\beta^{d-k}}{\prod_{i=k}^{d-1}R_{ii}}=\sqrt{\frac{1-\sum_{l=1}^{k-1}t_l^2}{t_d^2}}\cdot\beta^{d-k}\prod_{i=k}^{d-1}\sqrt{h_i^2v_i}$$
$$=\sqrt{1+\sum_{l=k}^{d-1}\frac{\sigma^2}{v_l}}\cdot\beta^{d-k}\prod_{i=k}^{d-1}\sqrt{h_i^2v_i}. \tag{60}$$

For ease of analysis, assume that the variances of check node messages v_i's are approximately the same if their corresponding h_i's are the same. Note that this approximation should hold accurately after the first few iterations. Because of the convolution (5) and stretching (6) steps, the variance of the i-th input check node message is

$$v_i=\frac{1}{h_i^2}\sum_{j=1,j\neq i}^{d}h_j^2\bar{V}_j, \tag{61}$$

where $\bar{V}_j$ is the variance of the variable node message with the moment matching approximation from the last iteration. According to (13) and (61),

$$v_i\geq\begin{cases}0 & \text{if } |h_i|=1\\ \frac{\sigma^2(1-\alpha)}{w^2} & \text{if } |h_i|=w\end{cases}. \tag{62}$$

Besides, note that $\beta\leq\beta_1$ due to (45). We then have

$$\beta^{d-k}\prod_{i=k}^{d-1}\sqrt{h_i^2v_i}\lessapprox\begin{cases}1 & \text{if } |h_k|=\cdots=|h_{d-1}|=w\\ \beta\sqrt{v_i} & \text{if } |h_i|=1 \text{ for some } i.\end{cases} \tag{63}$$

It is now prepared to decide the order of $|\mathcal{P}_k(\beta)|$ with the knowledge of the relationship between σ^2 and v_i. By combining with (58), (60), (62), and (63), two cases are considered depending on whether there is one check node message corresponding to $|h_i|=1$:

- When $|h_k|=\cdots=|h_{d-1}|=w$,

$$|\mathcal{P}_k(\beta)|\lessapprox\mathsf{V}_{d-k}\sqrt{1+(d-1)\frac{w^2}{1-\alpha}}\overset{(a)}{\sim}O(1), \tag{64}$$

  where (a) is due to $w=\sqrt{\frac{\alpha}{d-1}}$ and the fact that α is a constant.

- When $|h_k|=\cdots=|h_{i-1}|=|h_{i+1}|=\cdots=|h_{d-1}|=w$ and $|h_i|=1$,

$$|\mathcal{P}_k(\beta)|\lessapprox\mathsf{V}_{d-k}\sqrt{1+\frac{(d-2)w^2}{1-\alpha}+\frac{\sigma^2}{v_i}}\cdot\beta\sqrt{v_i}$$
$$\lessapprox\mathsf{V}_{d-k}\sqrt{\frac{\beta^2v_i}{1-\alpha}+\beta^2\sigma^2}$$
$$\lessapprox\mathsf{V}_{d-k}\sqrt{\frac{v_i}{\sigma^2(1-\alpha)^2}+\frac{\sigma^2}{\sigma^2(1-\alpha)}}. \tag{65}$$

  According to (61), for $|h_i|=1$, we have $v_i\leq(d-1)w^2\sigma^2$ with the initialization $\bar{V}_j=\sigma^2$. Thus,

$$|\mathcal{P}_k(\beta)|\lessapprox\mathsf{V}_{d-k}\sqrt{\frac{\alpha}{(1-\alpha)^2}+\frac{1}{1-\alpha}}\sim O(1). \tag{66}$$

By combining (64) and (66), the complexity of the LSD algorithm is

$$\sum_{k=1}^{d-1}|\mathcal{P}_k(\beta)|\sim O(d). \tag{67}$$

2) COMPLEXITY ORDER OF THE LPE DECODER

The complexity of computing a variable node message is dominated by that of the LSD algorithm. As mentioned in Section IV-C, the complexities for computing the inputs to the simplified LSD algorithm are all $O(d)$. According to (67), the search complexity of LSD algorithm is also $O(d)$. For computing the check node messages, the convolution of several Gaussian functions is equivalent to taking the sum of their means and variances, whose complexity is again $O(d)$. Thus, we conclude that the complexity order of the LPE decoder is $O(n \cdot t \cdot d)$. It is worth noting that LDLCs achieve slightly better error performance than the multilevel LDPC lattices [27] of the same dimensions. With linear complexity, the proposed LPE decoder further makes LDLCs competitive with the multilevel LDPC lattices.

B. ANALYSIS ON THE NUMBER OF OPERATIONS

Although the complexity order for the LPE decoder has been analyzed in Section V-A, for evaluating the complexity more explicitly, the comparison of floating-point operations (flops) between the LPE decoder and the M-Gaussian decoder [15] is presented in this subsection. A flop is assumed to include a real addition, subtraction, multiplication, or division. Besides, computing a square root needs 6 flops according to the IEEE floating-point representation [28] and 14 flops are needed for approximating an exponential function [29].

1) LPE DECODER

For one particular check node message, in the convolution step, calculating the mean needs $d-1$ multiplications and $d-2$ additions. In the stretching step, one more division is needed. Similarly, $2(d-1)$ multiplications and $d-2$ additions are needed to calculate the variance in the convolution step and two divisions are used in the stretching step. Thus, $5d-4$ flops are counted for each check node message.

For one particular variable node message, four d-dimensional vectors $\mathbf{t}, \mathbf{f}, \mathbf{g}, \mathbf{p}$, and the squared diagonal entries of $\mathbf{R}$ should be calculated first. The elements in these vectors can be either directly computed or iteratively obtained as mentioned before. Overall, computing these vectors needs $23d$ flops.

For the simplified LSD algorithm, 12 flops (from step 14 to step 16 in Algorithm 1) are required whenever a node is visited at one layer. Then, $\sum_{k=1}^{d-1} 12N_k$ flops are needed for the search where N_k is the number of visited nodes at the k-th layer. Then, for each candidate in $\mathcal{L}$, a Gaussian function is computed as described in (24), (25), and (28). Note that all Gaussian functions have the same variance which needs $2d$ flops and should be computed only once for all candidates. Besides, calculating the scaling coefficient is simple since the value of $(\mathbf{z}+\mathbf{h}\odot\mathbf{a})^T\mathbf{Q}(\mathbf{z}+\mathbf{h}\odot\mathbf{a})$ is already given in the list output of Algorithm 1. Taking the normalization of scaling coefficients into consideration, the number of flops for computing one variable node message are $12\sum_{k=1}^{d-1} N_k + (4d+15)L + 2d$ with L being the list size. Finally, the moment matching approximation needs $6L$ flops.

It is critical to emphasize that N_k and L are related to $\mathcal{P}_k(\beta)$ which is defined in Section V-A. Explicitly, we have $N_k = |\mathcal{P}_k(\beta)| + |\mathcal{P}_{k+1}(\beta)|$ and $L \le |\mathcal{P}_1(\beta)|$. From (64) and (66), N_k and L are $O(1)$. Therefore, the number of flops needed per variable node message is linear in d. In each variable node, the LPE decoder and the M-Gaussian decoder need to store L and M^{d-1} Gaussian functions, respectively. Since L is merely $O(1)$, the LPE decoder needs less storage than the M-Gaussian decoder.

2) *M*-GAUSSIAN DECODER

Similar to the LPE decoder, for one check node message, the convolution and stretching steps take $5d-4$ flops. Additional $3M+1$ flops are needed in the periodic extension.

For updating variable node messages, a forward-backward recursion is utilized in [15] for computing d messages sent from one variable node simultaneously. For the comparison, we first compute the number of flops needed for calculating d messages. Then, we use the average number of flops over d as the computational cost for one message. Given d input check node messages $p_l(x)$, the forward-backward recursion first computes two sequences of auxiliary messages $\{\theta_0(x), \ldots, \theta_{d-1}(x)\}$ and $\{\phi_2(x), \ldots, \phi_{d+1}(x)\}$, where

$$\theta_l(x) = \theta_{l-1}(x) \cdot p_l(x), \tag{68}$$

for $l = 1, \ldots, d-1$ with $\theta_0(x) = 1$, and

$$\phi_l(x) = \phi_{l+1}(x) \cdot p_l(x), \tag{69}$$

for $l = d, d-1, \ldots, 2$ with $\phi_{d+1}(x) = 1$. Then, each message sent from the k-th variable node is computed by multiplying two auxiliary messages as well as the channel message:

$$\tilde{f}_l(x) = \mathcal{N}\left(x; y_k, \sigma^2\right) \cdot \theta_{l-1}(x) \cdot \phi_{l+1}(x). \tag{70}$$

Counting the numbers of flops needed for the aforementioned calculations is trivial referring to (14)–(16). However, it should be mentioned that each auxiliary message is a Gaussian mixture, and normalization of scaling coefficients is always needed. Besides, the variances of all Gaussian functions of a mixture are the same, which should be computed only once. The number of flops for obtaining d variable node messages is $(65d-66)M^{d-1} + 66(M^d - M^2)/(M-1) + 5d^2 + 5d - 20$.

The numbers of flops needed per message for the LPE decoder and the M-Gaussian decoder [15] are summarized in Table 2.

VI. NUMERICAL RESULTS

Since the M-Gaussian decoder with a sufficiently large M achieves the best-known performance [15] and can be interpreted from the proposed lattice viewpoint, it is employed as the benchmark for the comparison with the LPE decoder in diverse aspects.

TABLE 2. The number of flops needed for the M-Gaussian and LPE decoders, where d, N_k and L are the degree of an LDLC, the number of visited nodes at the k-th layer and the list size, respectively.

The number of flops	Check node message	Inputs to LSD algorithm	Variable node message	Moment matching
M-Gaussian	$5d + 3M - 3$	–	$65M^{d-1} - 66d^{-1}M^{d-1}$ $+66d^{-1}(M^d - M^2)/(M-1)$ $+5d + 5 - 20d^{-1}$	$6 \cdot M^{d-1}$
LPE	$5d - 4$	$23d$	$12\sum_{k=1}^{d-1} N_k + (4d+15)L + 2d$	$6L$

A. NOISE THRESHOLD

The asymptotic performance of the LPE decoder is first evaluated by considering the noise threshold. However, the true density evolution needs the joint distribution of the means and variances of messages, which is computationally intractable for LDLCs. Alternatively, we perform the Monte Carlo density evolution [15], [30] for acquiring the noise threshold.

We first define two sets of messages $\mathcal{M}_{(1)}$ and $\mathcal{M}_{(w)}$. Messages in $\mathcal{M}_{(1)}$ correspond to $|h_i| = 1$ and those in $\mathcal{M}_{(w)}$ to $|h_i| = w$. The set $\mathcal{M}_{(1)}$ contains 10^5 messages while $\mathcal{M}_{(w)}$ has $(d-1)\cdot 10^5$ messages. All messages are denoted by means and variances. For initialization, all means of messages in both sets are randomly generated from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$ and all variances are set to be σ^2.

For the inputs of check/variable nodes, one message is drawn from $\mathcal{M}_{(1)}$ and $d-1$ messages are taken from $\mathcal{M}_{(w)}$, all randomly. The calculation of check/variable node messages follows decoding steps in the LPE decoder. The outputs of check/variable nodes will be the updated message sets for variable/check nodes iteratively. When the mean of message variances in $\mathcal{M}_{(w)}$ is below 0.001 within 50 iterations, the convergence is declared [15], [30].

As reported in [15], increasing M beyond 3 provides no visible improvement in the error performance. We only compare the noise thresholds of LDLCs employing our decoder with those using the 3-Gaussian decoder, as indicated in Fig. 2. Since the value of d is commonly set to 7 [4], we also set $d = 7$ in the simulation. The noise threshold provides the limit of error performance of a decoder when the code length tends to infinity. As will be shown in Section VI-B, the gaps from the capacity of different decoders approach to the noise thresholds indicated in Fig. 2. The lowest noise threshold is 0.64 dB with $\alpha = 0.75$. The 3-Gaussian decoder and the LPE decoder achieve almost the same noise thresholds.

B. FINITE LENGTH PERFORMANCE SIMULATION

Assuming that no power constraint is considered, the symbol error rate (SER) performance for different code parameters is shown in Fig. 3. Note that a symbol error is declared if $\hat{b}_i \neq b_i$ for $i = 1, \ldots, n$. The generating sequences are $\left\{1, 1/\sqrt{3}, 1/\sqrt{3}\right\}$ for $n = 10^2$ with $d = 3$, and $\left\{1, 1/\sqrt{7}, \ldots, 1/\sqrt{7}\right\}$ for $n = 10^3$, and 10^4 with $d = 7$, respectively. Simulation results are obtained by assuming that the all-zero codeword is transmitted through the AWGN channel and the maximum number of iterations is set to 100.

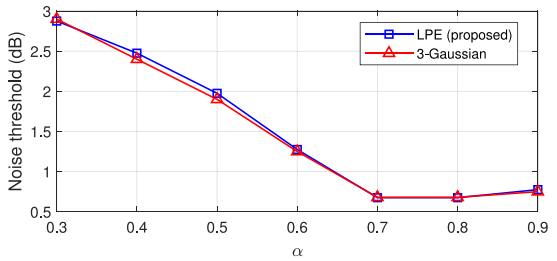


FIGURE 2. Noise thresholds of LDLCs with $d = 7$ measured by VNR using the LPE decoder and the 3-Gaussian decoder.

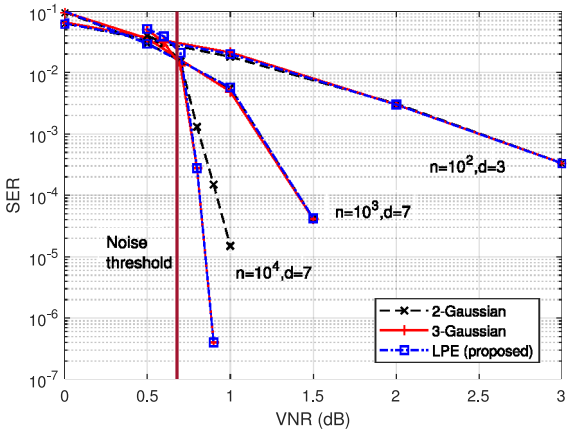
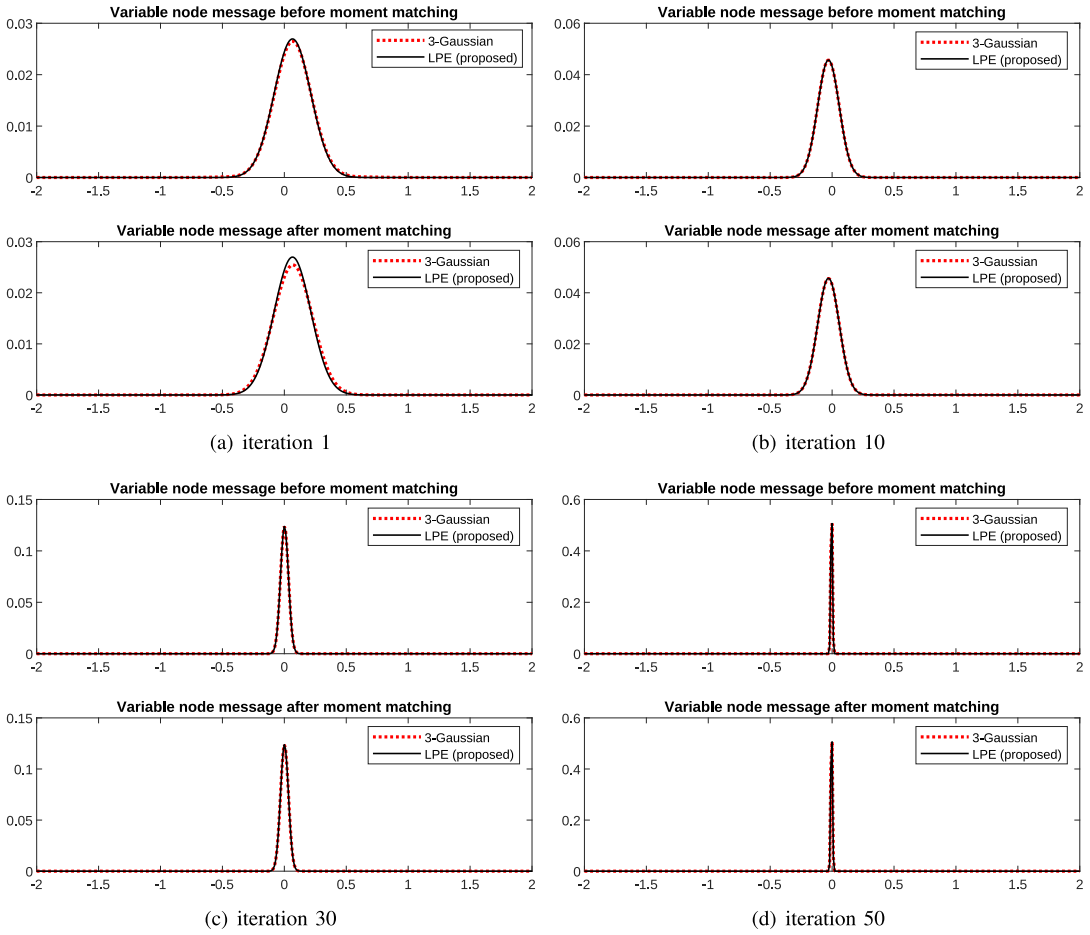


FIGURE 3. SER performance comparison between the M-Gaussian decoder with M= 2, 3 and the LPE decoder for different LDLC parameters.

TABLE 3. Probability of an empty list output for the LSD algorithm at the first iteration with $n = 10^4$, $d = 7$ and $w = 1/\sqrt{7}$.

VNR (dB)	Probability of an empty list
0.7	9.857×10^{-6}
0.8	1.028×10^{-5}
0.9	9.286×10^{-6}

As illustrated in Fig. 3, the SER performance is almost the same for $n = 10^2$ and $n = 10^3$ when applying the LPE decoder and the M-Gaussian decoder. However, for $n = 10^4$, the LPE decoder outperforms 0.15 dB over the 2-Gaussian decoder for the SER of 10^{-5} and still achieves almost the same performance as the 3-Gaussian decoder. The 2-Gaussian decoder may miss some essential Gaussian functions for approximating variable node messages, whereas the LPE decoder achieves better approximation. As a consequence, the LPE decoder provides better SER performance than the 2-Gaussian decoder.



(a) iteration 1 (b) iteration 10

(c) iteration 30 (d) iteration 50

FIGURE 4. Message comparison between the 3-Gaussian and LPE decoders for the non-empty list case: messages are shown before and after the moment matching at different iterations.

C. PROBABILITY EVALUATION FOR EMPTY LIST OUTPUT OF LSD ALGORITHM

As mentioned in Section IV-B, the LSD algorithm may output an empty list at a low probability. We evaluate the probability by decoding 100 noisy codewords with parameters $n = 10^4$ and $d = 7$. Note that each variable node should generate d messages. Thus, there are 7×10^6 samples under evaluation in total. The empty list output is only found at the first iteration. The variable node message will not be updated if an empty list is output. We provide the probability of an empty list output at the first iteration in Table 3. The probability of an empty list is around 10^{-5}. In other words, for decoding one codeword, the message updating could be delayed at most by one iteration at a probability of 10^{-5}.

D. MESSAGE VISUALIZATION

For examining the approximation accuracy, the variable node messages in the LPE decoder are visualized in this subsection, as well as those obtained by applying the 3-Gaussian decoder. For different iterations, messages in the 3-Gaussian decoder and the LPE decoder are compared for the same pair of variable and check nodes. Since the LPE decoder may encounter an empty list output at a probability of 10^{-5}, the comparison is shown for both non-empty list case and empty list case.

The message visualization for the non-empty list case is depicted in Fig. 4. Messages before and after the moment matching approximation are visualized. At the first iteration, the messages in two decoders are slightly different but become almost the same with the iteration going on. The similarity of messages for the 3-Gaussian and the LPE decoder indicates that our decoder design has almost no degradation in terms of approximation accuracy. The consistent results of noise thresholds and error performance for the 3-Gaussian and the LPE decoder in previous subsections are further validated.

The message comparison for the empty list case is shown in Fig. 5. At the first iteration, the message in the LPE decoder is set to be the channel message since it cannot be updated as shown in Fig. 5(a). Although messages in two decoders at early iterations mismatch with each other in terms of their means and variances, they still become similar after sufficient iterations. In other words, the effect caused by the empty list at the first iteration can be remedied as long as the number of iterations is sufficiently large.

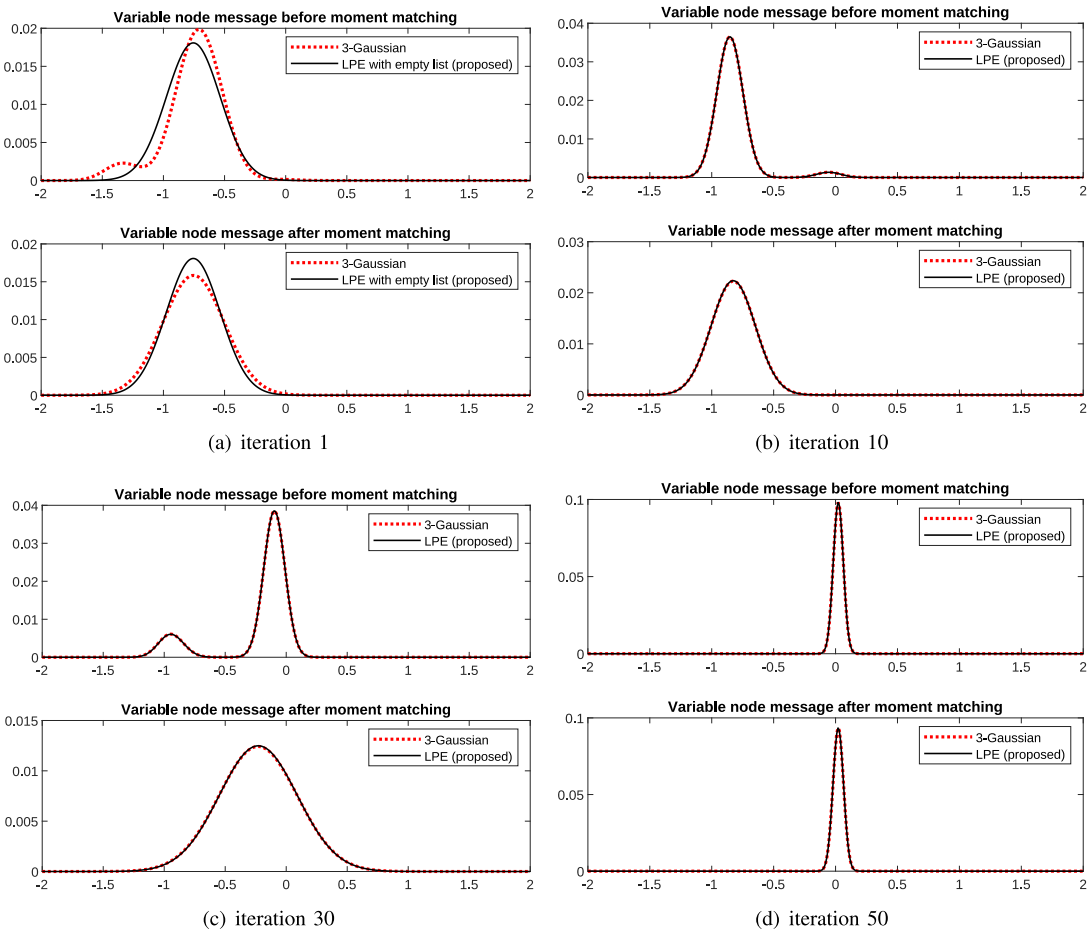


FIGURE 5. Message comparison between the 3-Gaussian and LPE decoders for the empty list case: messages are shown before and after the moment matching at different iterations.

E. COMPARISON ON THE NUMBER OF FLOPS

As analyzed in Section V-B, the number of flops needed to compute a variable node message depend on the number of visited nodes N_k and the list size L of the simplified LSD algorithm. Since the number of visited nodes at each layer and the list size may vary when computing different variable node messages, the average values of N_k and L are considered [31]. We first evaluate the average number of visited nodes and that of the list size per variable node message numerically. Then, we calculate the average number of flops needed per message for both the M-Gaussian decoder and the LPE decoder.

The average list sizes at different iterations are depicted in Fig. 6 for $n = 10^3$ and 10^4 with different values of VNR. The maximum average list size does not surpass 13 for all considered cases, which potentially indicates that the best-known performance could be achieved by considering a fewer number of Gaussian functions than that used in the M-Gaussian decoder. Besides, the trend of the average list size matches well with the inherent property of the message passing decoder. At early iterations, the average list size increases since the decoder tries to approximate every variable node message with a maximum number of Gaussian functions. After reaching the peak, the list size starts decreasing because

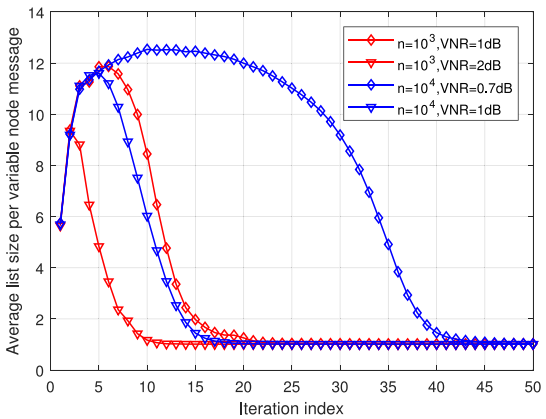
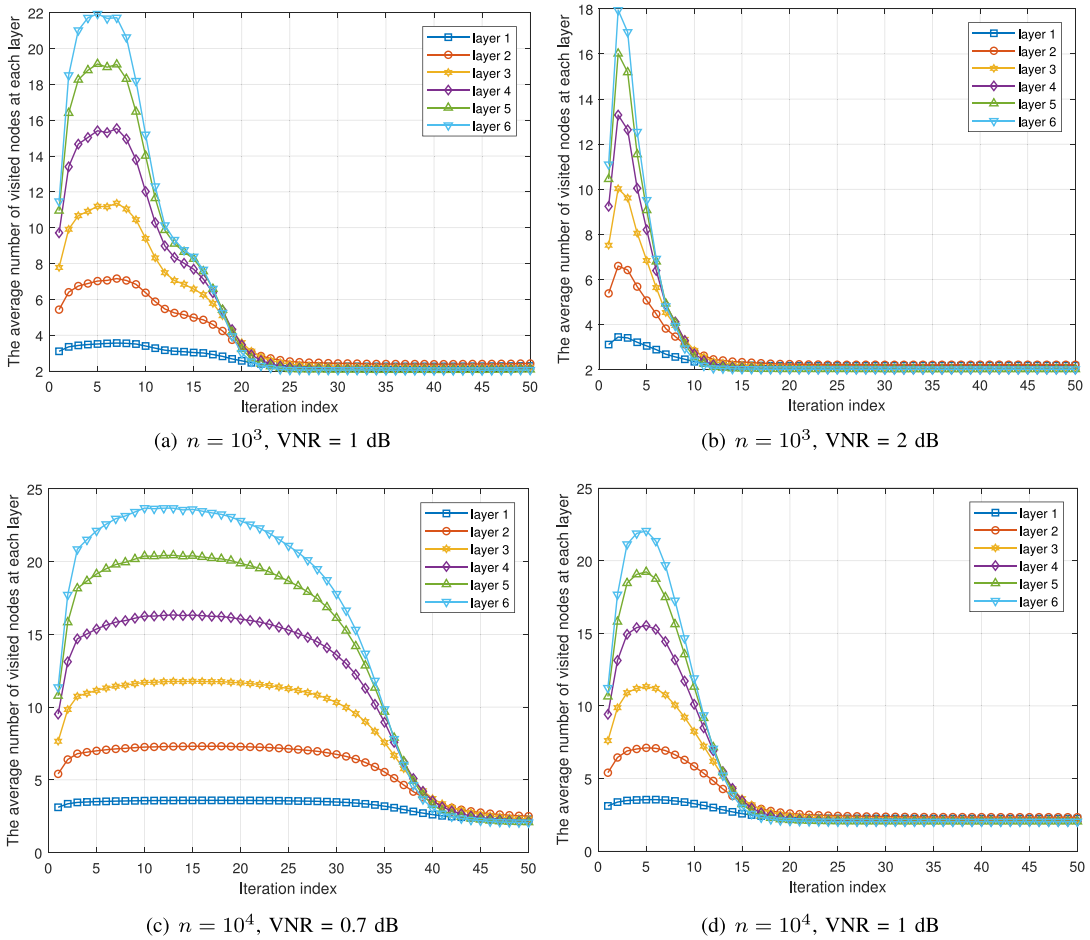


FIGURE 6. Average list size for $n = 10^3$, 10^4 and $d = 7$ at both low and high VNR in different iterations.

of the convergence property of the decoder. Particularly, the list size becomes one when the convergence is declared, which means that only one Gaussian function is dominant for approximating each variable node message. Besides, the convergence speed of the list size is sensitive to the value of VNR. It takes only 10 iterations to reach the convergence at high VNR (2 dB), but needs more than 40 iterations to declare the convergence at low VNR (0.7 dB).



(a) $n = 10^3$, VNR = 1 dB

(b) $n = 10^3$, VNR = 2 dB

(c) $n = 10^4$, VNR = 0.7 dB

(d) $n = 10^4$, VNR = 1 dB

FIGURE 7. The average number of visited nodes at each layer for $n = 10^3, 10^4$ and $d = 7$ at both low and high VNR in different iterations (layer 1 is the root layer and layer 6 is the leaf layer).

The average number of visited nodes at each layer is shown in Fig. 7 by simulation with $n = 10^3, 10^4$ and $d = 7$. Since the number of visited nodes may vary with iterations and values of VNR, the first 50 iterations are considered at both low and high VNR. It is observable that the trend of the average number of visited nodes is consistent with that of the average list size.

For the M-Gaussian decoder, the number of flops needed per variable node message is a constant regardless of the iteration index as given in Table 2. While for the LPE decoder, the number of flops is related to the average number of visited nodes and list size which both vary with iterations. For this reason, the maximum average number of visited nodes and list size over different iterations are considered to compute the number of flops needed for the LPE decoder. In Fig. 8, we show the average numbers of flops needed per variable node message for the LPE decoder and the M-Gaussian decoders with $M = 2$ and 3. For $d \geq 4$, it is shown that the LPE decoder needs fewer flops on average than the M-Gaussian decoders. In particular, the number of flops of the LPE decoder is 33.9% of that of 2-Gaussian decoder and only 3.2% of that of 3-Gaussian decoder when $d = 7$.

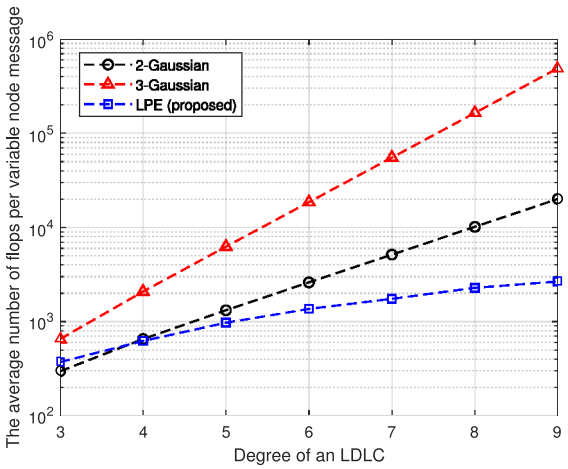


FIGURE 8. Comparison between the M-Gaussian decoder and the LPE decoder in terms of the average number of flops needed per variable node message with $n = 10^3$ and VNR = 1 dB.

F. RUNTIME COMPARISON

The runtime comparison between the M-Gaussian decoder and the LPE decoder is shown in Fig. 9 using MATLAB 2017b on a single computer, with an Intel Core i5-6500

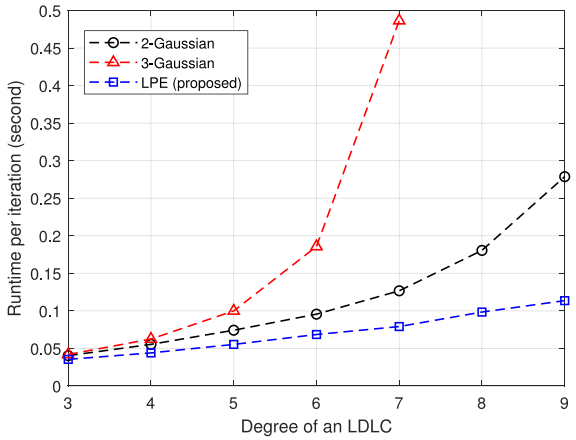


FIGURE 9. Runtime comparison for different decoders with $n = 10^3$ and VNR = 1 dB.

CPU, a RAM of 8 GB, and Windows 10 Enterprise operating system. By choosing $n = 10^3$, the exponential and linear complexity orders in d for two decoders are confirmed in Fig. 9, respectively. It is obvious that the LPE decoder always has less runtime than the 3-Gaussian decoder and becomes faster than the 2-Gaussian decoder when d is larger than 5. The result of the runtime comparison is consistent with that of the comparison on the number of flops in Section VI-E. Specifically, for $d = 7$, the LPE decoder achieves around (1-0.0791/0.4865)×100% = 83.7% runtime saving compared to the 3-Gaussian decoder.

VII. CONCLUSION

In this paper, an efficient LPE decoder for LDLCs has been proposed from the lattice viewpoint. For approximating the variable node messages in the form of Gaussian mixtures, each Gaussian function is first related to a lattice point in a specific lattice. A simplified LSD algorithm is then derived for efficiently enumerating lattice points corresponding to essential Gaussian functions. Compared to the M-Gaussian decoder whose complexity is $O(n \cdot t \cdot M^{d-1})$, the LPE decoder has only linear complexity $O(n \cdot t \cdot d)$. Nevertheless, the LPE decoder still achieves almost the same noise threshold and error performance as the M-Gaussian decoder. As an example, for $n = 1000$ and $d = 7$, compared to the 3-Gaussian decoder, the proposed LPE decoder reduces the number of flops and runtime by 96.8% and 83.7%, respectively.

APPENDIX
DERIVATION OF (28) FROM (26)

Define $\hat{a}_i \triangleq a_i + z_{s,i}/h_i$. The exponent of (26) becomes

$$\begin{aligned} & V \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{(\hat{a}_i - \hat{a}_j)^2}{v_i v_j} \\ &= \frac{V}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{(\hat{a}_i - \hat{a}_j)^2}{v_i v_j} \\ &= V \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\hat{a}_i^2}{v_i v_j} - V \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\hat{a}_i \hat{a}_j}{v_i v_j}. \end{aligned} \tag{71}$$

Let $\hat{\mathbf{a}} = [\hat{a}_1, \ldots, \hat{a}_d]^T$. The first term in the right-hand side of (71) can be written as

$$\begin{aligned} V \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\hat{a}_i^2}{v_i v_j} &\overset{(b)}{=} \sum_{i=1}^{d} \frac{\hat{a}_i^2}{v_i} \\ &= \hat{\mathbf{a}}^T \mathrm{diag}([1/v_1, \ldots, 1/v_d])\hat{\mathbf{a}} \\ &= (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_1 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}), \end{aligned} \tag{72}$$

where (b) is due to (24). The second term in the right-hand side of (71) is

$$\begin{aligned} V \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\hat{a}_i \hat{a}_j}{v_i v_j} &= \hat{\mathbf{a}}^T \begin{bmatrix} \frac{V}{v_1 v_1} & \cdots & \frac{V}{v_1 v_d} \\ \vdots & \ddots & \vdots \\ \frac{V}{v_d v_1} & \cdots & \frac{V}{v_d v_d} \end{bmatrix} \hat{\mathbf{a}} \\ &= (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q}_2 (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a}). \end{aligned} \tag{73}$$

Since $\mathbf{Q} = \mathbf{Q}_1 - \mathbf{Q}_2$, the scaling coefficient (26) becomes $c_s = Ce^{-\frac{1}{2}(\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})^T \mathbf{Q} (\mathbf{z}_s + \mathbf{h} \odot \mathbf{a})}$.

REFERENCES

[1] X. Wang and W. H. Mow, "Efficient LDLC decoder design from the lattice viewpoint," in *Proc. IEEE GLOBECOM*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[2] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[3] 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, no. 10, pp. 2293–2314, Oct. 2006.

[4] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1561–1585, Apr. 2008.

[5] Y. Wang and A. Burr, "Physical-layer network coding via low density lattice codes," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2014, pp. 1–5.

[6] Y. Wang, A. Burr, and D. Fang, "Complex low density lattice codes to physical layer network coding," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 2060–2065.

[7] B. Chen, D. N. Jayakody, and M. F. Flanagan, "Low-density lattice coded relaying with joint iterative decoding," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 4824–4837, Dec. 2015.

[8] N. S. Ferdinand, M. Nokleby, and B. Aazhang, "Low-density lattice codes for full-duplex relay channels," *IEEE Trans. Wireless Commun.*, vol. 14, no. 4, pp. 2309–2321, Apr. 2015.

[9] B. Chen, D. N. Jayakody, and M. F. Flanagan, "Distributed low-density lattice codes," *IEEE Commun. Lett.*, vol. 20, no. 1, pp. 77–80, Jan. 2016.

[10] Y. Tan, S. C. Liew, and T. Huang, "Mobile lattice-coded physical-layer network coding with practical channel alignment," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1908–1923, Aug. 2018.

[11] X. Wang and W. H. Mow, "Decoding and convergence analysis for distributed low density lattice codes," in *Proc. 17th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Berlin, Germany, Sep. 2021, pp. 1–6.

[12] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[13] B. Kurkoski and J. Dauwels, "Message-passing decoding of lattices using Gaussian mixtures," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2008, pp. 2489–2493.

[14] Y. Yona and M. Feder, "Efficient parametric decoder of low density lattice codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun./Jul. 2009, pp. 744–748.

[15] R. A. P. Hernandez and B. M. Kurkoski, "The three/two Gaussian parametric LDLC lattice decoding algorithm and its analysis," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3624–3633, Sep. 2016.

[16] W. Wiriya and B. M. Kurkoski, "Reliability-based parametric LDLC decoding," in *Proc. Int. Symp. Inf. Theory Appl. (ISITA)*, Oct. 2018, pp. 188–192.
[17] S. Liu, Y. Hong, E. Viterbo, A. Marelli, and R. Micheloni, "Efficient decoding of low density lattice codes," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1195–1199, Aug. 2019.
[18] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, 1981.
[19] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.
[20] R. Reams, "Hadamard inverses, square roots and products of almost semidefinite matrices," *Linear Algebra Appl.*, vol. 288, pp. 35–43, Feb. 1999.
[21] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 409–417, Mar. 1994.
[22] C.-P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, nos. 1–3, pp. 181–199, 1994.
[23] B. Kurkoski and J. Dauwels, "Reduced-memory decoding of low-density lattice codes," *IEEE Commun. Lett.*, vol. 14, no. 7, pp. 659–661, Jul. 2010.
[24] J. Wen, B. Zhou, W. H. Mow, and X.-W. Chang, "An efficient algorithm for optimally solving a shortest vector problem in compute-and-forward design," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6541–6555, Oct. 2016.
[25] C. Ling and J.-C. Belfiore, "Achieving AWGN channel capacity with lattice Gaussian coding," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5918–5929, Oct. 2014.
[26] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.
[27] P. R. B. da Silva and D. Silva, "Multilevel LDPC lattices with efficient encoding and decoding and a generalization of construction D′," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 3246–3260, May 2019.
[28] J. F. Blinn, "Floating-point tricks," *IEEE Comput. Graph. Appl.*, vol. 17, no. 4, pp. 80–84, Jul. 1997.
[29] T. Ahrendt, "Fast computations of the exponential function," in *Proc. Annu. Symp. Theor. Aspects Comput. Sci.*, 1999, pp. 302–312.
[30] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Single-Gaussian messages and noise thresholds for decoding low-density lattice codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2009, pp. 734–738.
[31] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

XUEBO WANG (Student Member, IEEE) received the B.Eng. degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2016. He is currently pursuing the Ph.D. degree in electronic and computer engineering with the Hong Kong University of Science and Technology, Hong Kong, China. His research interests include lattice coding theory, message passing decoder design, and low-power bus coding.

WAI HO MOW (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in 1993. From 1997 to 1999, he was with Nanyang Technological University, Singapore. Since 2000, he has been with the Hong Kong University of Science and Technology, where he is currently a Professor. He unified all known constructions of perfect roots-of-unity (also known as CAZAC) sequences, which have been widely used as communication preambles and radar signals. He published two books and 220+ journal/conference publications and is the inventor of 38 patents. He pioneered the lattice approach to signal detection problems, including sphere decoding and complex lattice reduction-aided detection. His research areas include coding and information theory, wireless communications, optical camera communications, and thermographic signal processing. His joint work won the top prizes of 10+ project/paper competitions, including the 2014 HK U-21 IoT Gold Award for Revolutional Concept, the Best Paper Award of 2013 and 2016 Asia–Pacific Communications Conference, and the Best Mobile App Award at ACM MobiCom'2013. His co-invented picture barcode PiCode was highlighted as one of the four local innovations in the 2015 International IT Fest organized by the Office of the Government Chief Information Officer, Hong Kong. He served on the editorial boards of six journals, including the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He is the Past Chair of the Hong Kong Chapter of the IEEE Information Theory Society, and was the General/Program Chair of six conferences, including SETAŠ2018 held in Hong Kong. He is a Past Member of the Radio Spectrum Advisory Committee, Office of the Telecommunications Authority of the Hong Kong SAR Government.