Message-Passing Decoding of Lattices Using Gaussian Mixtures

Brian Kurkoski
University of Electro-Communications
Tokyo, Japan
kurkoski@ice.uec.ac.jp

Justin Dauwels
Massachusetts Institute of Technology, Cambridge, MA
Harvard Medical School, Boston, MA
justin@dauwels.com

Abstract— **A belief-propagation decoder for low-density lattice codes, which represents messages explicitly as a mixture of Gaussians functions, is given. In order to prevent the number of functions from growing as the decoder iterations progress, a method for reducing the number of Gaussians at each step is given. A squared distance metric is used, which is shown to be a lower bound on the divergence. For an unconstrained power system, comparisons are made with a quantized implementation. For a dimension 100 lattice, a loss of about 0.2 dB was found; for dimension 1000 and 10000 lattices, the difference in error rate was indistinguishable. The memory required to store the messages is substantially superior to the quantized implementation.**

I. Introduction

Recently, a new lattice construction and decoding algorithm, based upon the ideas of low-density parity check codes has been introduced. So-called low-density lattice codes (LDLC) are lattices defined by sparse inverse generator matrix, and are decoded using a belief-propagation algorithm, with complexity that is linear in the lattice dimension. Sommer, Feder and Shalvi, who proposed this lattice and decoder, decoded a lattice with dimension 10^6, and a noise threshold appeared within 0.6 dB of the channel capacity of a system which does not constrain the transmit power [1].

In belief-propagation decoding of LDLC codes the messages are continuous functions. When the channel is AWGN, these messages are a mixture of Gaussian functions. This is appealing for a decoder implementation, but unfortunately as the iterations progress the number of Gaussians in the mixture grows quite rapidly, and a naive implementation using Gaussians is infeasible. Thus, in prior work, the decoder messages were quantized. In Section II, the assumed communications system, LDLC codes, and the Gaussian-mixture belief-propagation decoder are reviewed.

This paper describes an implementation of the belief-propagation LDLC decoder which represents the messages as mixtures of Gaussians. The key component is an algorithm which approximates a Gaussian mixture by a smaller number of Gaussians.

B. Kurkoski was supported in part by the Ministry of Education, Science, Sports and Culture; Grant-in-Aid for Scientific Research (C) number 19560371. J. Dauwels was supported in part by post-doctoral fellowships from the King Baudouin Foundation and the Belgian American Educational Foundation (BAEF). Part of this work was carried out at the RIKEN Brain Science Institute, Saitama, Japan.

The proposed algorithm compares the distance between all possible pairs of Gaussians in an input list, and combines the closest two by replacing them with a single Gaussian. This proceeds iteratively until a stopping condition is reached. While the Kullback-Leibler (KL) divergence would be an appropriate distance measure, it cannot be found in closed form, and a lower bound on the KL divergence is used instead. This lower bound is the squared distance, can be computed in closed form, and thus is used as the distance metric. The two Gaussians are replaced by a single Gaussian using the method of moments, which minimizes the KL divergence. This algorithm, which we refer to as the Gaussian mixture reduction algorithm, is described in Section III.

A related technique is the iterative pairwise replacement algorithm for reducing the order of a Gaussian mixture in kernel density estimation [2]. For compressed sensing, this algorithm was applied to find the solution to an underdetermined system of equations, when there is an a priori distribution on the unknowns [3]. This inference algorithm has some similarities with LDLC decoding.

The Gaussian mixture reduction algorithm is then applied to the LDLC belief-propagation decoder. Minor refinements of the belief-propagation algorithm are required for efficient implementation. By numerical evaluation, the proposed algorithm has error-rate performance very close to the quantized implementation. While it is difficult to compare the computational complexity of the two algorithms, the proposed technique uses substantially less memory. The proposed LDLC decoder and the complexity considerations are discussed in Section IV.

II. Low-density Lattice Codes

A. Lattices and the Unconstrained Power System

An n-dimensional lattice Λ is defined by an n-by-n generator matrix G. The lattice consists of the discrete set of points $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ with

$$\mathbf{x} = G\mathbf{b}, \qquad (1)$$

where $\mathbf{b} = (b_1, \ldots, b_n)$ is the set of all possible integer vectors, $b_i \in \mathbb{Z}$. Lattices are linear, in the sense that $\mathbf{x}_1 + \mathbf{x}_2 \in \Lambda$ if $\mathbf{x}_1$ and $\mathbf{x}_2$ are lattice points.

We consider the unconstrained power system, as was used by Sommer, et al. Let the codeword $\mathbf{x}$ be an arbitrary point of the lattice Λ. This codeword is transmitted over an AWGN channel, where noise z_i with noise variance σ^2 is added to each code symbol. Then the received sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ is:

$$y_i \quad = \quad x_i + z_i, i = 1, 2, \ldots, n. \tag{2}$$

A maximum-likelihood decoder selects $\widehat{\mathbf{x}}$ as the estimated codeword:

$$\widehat{\mathbf{x}} \quad = \quad \arg\max_{\mathbf{x}\in\Lambda} Pr(\mathbf{y}|\mathbf{x}). \tag{3}$$

The received codeword is correct if $\mathbf{x} = \widehat{\mathbf{x}}$ and incorrect otherwise.

Notably, the transmit power $||\mathbf{x}||^2$ is unbounded. Instead, the system is constrained by the codeword density, measured by the volume of the lattice's Voronoi region, given by $|\det(G)|$. Poltyrev showed [4] that for sufficiently large n, there exists a lattice for which the probability of error becomes arbitrarily small, if and only if,

$$\sigma^2 \quad \leq \quad \frac{|\det(G)|^{2/n}}{2\pi e}. \tag{4}$$

Significantly, Poltyrev's work allows for separation of the design of the lattice from the design of some shaping region which enforces the power constraint. This setting is convenient for examining LDLC codes, since the code, which is the same as the lattice, is linear, and the absence of a power constraint simplifies the decoder.

B. LDLC Codes

A *low-density lattice code* is a lattice with a non-singular generator matrix G, for which $H = G^{-1}$ is sparse. It is convenient to assume that $\det(H) = 1/\det(G) = 1$. An (n, d) *regular LDLC code* has an H matrix with constant row and column weight d. In a *magic square LDLC*, the values of the d non-zero coefficients in each row and each column have the values $h_1, h_2, \ldots, h_d$. For convenience, these values are sorted as $h_1 \geq h_2 \geq \cdots \geq h_d > 0$. The signs of these entries of H are pseudo-randomly changed to minus with probability one-half.

C. LDLC Belief-Propagation Decoder

The LDLC belief-propagation decoding algorithm estimates the a posteriori probability $Pr(x_i|\mathbf{y})$. As with decoding low-density parity check codes, the decoding algorithm may be presented on a bipartite graph. For regular LDLC codes, there are nd variable-to-check messages $q_k(z)$, and nd check-to-variable messages $r_k(z)$, $k = 1, 2, \ldots, nd$. Since the variables x_i are continuous, the messages are functions.

A Gaussian with mean m and variance v is denoted as:

$$\mathcal{N}(z; m, v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(z-m)^2}{2v}}. \tag{5}$$

Because the channel noise is Gaussian, all messages are mixtures of Gaussians. The message $f(z)$ is a mixture of N Gaussians:

$$f(z) = \sum_{i=1}^{N} c_i \mathcal{N}(z; m_i, v_i)\,, \tag{6}$$

where $c_i \geq 0$ are the mixing coefficients with $\sum_{i=1}^{N} c_i = 1$. In this way, the message $f(z)$ can be described by a list of triples of means, variances and mixing coefficients, $\{(m_1, v_1, c_1), \ldots, (m_N, v_N, c_N)\}$

For the AWGN channel, node i has channel output y_i, and channel message is $y_i(z) = \mathcal{N}\big(z; y_i, \sigma^2\big)$ (we distinguish between y_i and $y_i(z)$). The initial variable-to-check message for edge k is $q_k(z) = y_i(z)$, if edge k is connected to variable node i.

In describing the Gaussian mixture decoder, initially assume that the input messages to the variable node and check node consist of a single Gaussian. That is $N = 1$ and $q_k(z)$ and $r_k(z)$ are of the form $\mathcal{N}(z; m_k, v_k)$.

Check node Without loss of generality, consider check node inputs $k = 1, \ldots, d-1$ and output d. The corresponding non-zero coefficients from H are $h_1, \ldots, h_d$. The output $r_d(z)$ is:

$$r_d(z) \;=\; \sum_{b=-\infty}^{\infty} \mathcal{N}(z; m_d(b), v_d)\,, \tag{7}$$

where,

$$m_d(b) \quad = \quad -\frac{\sum_{k=1}^{d-1} h_k m_k + b}{h_d}, \text{ and} \tag{8}$$

$$v_d \quad = \quad \frac{\sum_{k=1}^{d-1} h_k^2 v_k}{h_d^2}. \tag{9}$$

Variable Node. Similarly, consider the variable node inputs $k = 1, \ldots, d-1$ and output d. For notational convenience, let the channel message be $r_0(z) = \mathcal{N}(z; m_0, v_0)$, with $m_0 = y_i$, the received symbol at node i, and $v_0 = \sigma^2$, the channel variance. The output message $q_d(z)$, the product of these input messages, will also be a Gaussian,

$$q_d(z) \quad = \quad k_d \mathcal{N}(z; m_d, v_d)\,, \tag{10}$$

where,

$$\frac{1}{v_d} = \sum_{k=0}^{d-1} \frac{1}{v_k}, \qquad \frac{m_d}{v_d} = \sum_{k=0}^{d-1} \frac{m_k}{v_k}, \tag{11}$$

and,

$$k_d = \sqrt{\frac{v_d}{(2\pi)^{d-2}\prod_i v_i}} \exp\Big(-\frac{v_d}{2}\sum_{i=0}^{d-2}\sum_{j=i+1}^{d-1}\frac{(m_i - m_j)^2}{v_i v_j}\Big). \tag{12}$$

For the general case where the input consists of a mixture of Gaussians, each element of the output mixture can be found by conditioning on one element from each input mixture and performing the above computations, at both the check node and the variable node.

Consider the variable node, with inputs $r_k(z)$ which are a mixture of N_k Gaussians, $(m_k^{(i)}, v_k^{(i)}, c_k^{(i)})$ for $k = 0, \ldots, d-1$ and $i = 1, 2, \ldots, N_k$ ($N_0 = 1$, the channel message). For each input k, condition on one Gaussian $i_k \in \{1, \ldots, c_k\}$. This contributes one Gaussian to the output mixture $q_d(z)$, where the mean and variance is computed from (11), using $m_k^{(i_k)}$ and $v_k^{(i_k)}$. The mixing coefficient for this Gaussian is $k_d \prod_{k=1}^{d-1} c_k^{(i_k)}$, where k_d is given by (12). The check node output can be computed in a similar way.

The number of Gaussians in each mixture grows rapidly as the iterations progress. In practice, the sum in eqn. (7) may be taken over some set of I integers. On any iteration, if each check node input is a mixture of N Gaussians, then the output will be a mixture of IN^{d-1} Gaussians. A similar argument is made at the variable node, so that the check node input on the following iteration is a mixture of $I^{d-1}N^{(d-1)^2}$ Gaussians. The initial mixture consists of a single Gaussian, and so the number of Gaussians in the check node input mixture after iteration i is greater than $I^{(d-1)^{2i-1}}$. Thus, a naive implementation of this Gaussian mixture decoder is prohibitively complex.

III. Gaussian Mixture Reduction Algorithm

In Section III-A, a method which finds a single Gaussian which is a good approximation of a mixture of two Gaussians, is described. Further, to evaluate the goodness of the approximation, a distance metric between these two distributions is given. In Section III-B, the proposed Gaussian mixture reduction algorithm is described, which uses this distance metric to reduce a mixture of an arbitrary number of N Gaussians to a mixture of a smaller number of Gaussians.

A. Approximating a Mixture of Two Gaussians with a Single Gaussian, and Their Distance

When approximating a true distribution $p(z)$, by an approximate distribution $q(z)$, a reasonable approach is to choose $q(z)$ in such a way that the Kullback-Leibler divergence $\mathrm{KL}(p\|q)$ is minimized. For distributions with support $\mathcal{Z}$, the KL divergence is given by:

$$\mathrm{KL}(p\|q) = \int_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)}\, dz. \tag{13}$$

When $q(z)$ is a Gaussian, selecting the mean and variance to be the same as those of $p(z)$ will minimize the divergence; this is sometimes known as "moment matching." In particular, if $p(z)$ is a mixture of two Gaussians we have the following.

Lemma 1: The single Gaussian with mean m and variance v which minimizes the divergence from the mixture of two Gaussians $c_1 \mathcal{N}(z; m_1, v_1) + c_2 \mathcal{N}(z; m_2, v_2)$ is given by:

$$\begin{aligned} m &= c_1 m_1 + c_2 m_2, \text{ and} && (14)\\ v &= c_1(m_1^2 + v_1) + c_2(m_2^2 + v_2) \\ &\quad - c_1^2 m_1^2 - 2c_1c_2 m_1 m_2 - c_2^2 m_2^2. && (15) \end{aligned}$$

For convenience, we use the following notation. Let $t_i, i = 1, 2$ denote the triple (m_i, v_i, c_i), where $c_1 + c_2 = 1$. The single Gaussian which satisfies the property of Lemma 1 is denoted as:

$$t = \mathrm{MM}(t_1, t_2), \tag{16}$$

where $t = (m, v, 1)$, with m and v as given in (14) and (15).

While it is easy to find the mean and variance of the single Gaussian $q(z)$ which minimizes the divergence, the divergence itself does not appear to have a closed form. Computing the divergence numerically is complex. Instead, we compute a lower bound on the divergence, the squared difference.

Definition The squared difference $\mathrm{SD}(p\|q)$ between two distributions $p(z)$ and $q(z)$ with support $\mathcal{Z}$ is defined as:

$$\mathrm{SD}(p\|q) \quad = \quad \int_{z \in \mathcal{Z}} (p(z) - q(z))^2 dz. \tag{17}$$

The squared difference is non-negative and zero if and only if $p = q$. The squared difference is symmetric.

For discrete distributions with support $\mathcal{A}$, the squared difference forms a lower bound on the divergence. The $\mathcal{L}_1$ distance between $p(a)$ and $q(a)$ is $\sum_{a \in \mathcal{A}} |p(a) - q(a)|$. The divergence is lower bounded by [5, Sec. 12.6]:

$$\mathrm{KL}(p\|q) \quad \geq \quad \frac{1}{2 \ln 2} \Big(\sum_{a \in \mathcal{A}} |p(a) - q(a)| \Big)^2. \tag{18}$$

Expanding the square on the right-hand side of (18), the sum of the squares $|p(a) - q(a)|^2$ is equal to $\mathrm{SD}(p\|q)$, and the cross terms are all non-negative. Thus, the $\mathcal{L}_1$ distance is lower bounded by the squared distance and we have proved the following.

Lemma 2: For any two discrete distributions p and q:

$$\mathrm{KL}(p\|q) \quad \geq \quad \frac{1}{2 \ln 2} \mathrm{SD}(p\|q). \tag{19}$$

It is reasonable to assume that Lemma 2 holds for well-behaved continuous functions as well.

Computing the squared distance between two Gaussians and a single Gaussian is tractable (computing the $\mathcal{L}_1$ distance requires evaluation of error functions, and is not considered for complexity reasons). In Section III-B, it will be convenient to have a function which describes the penalty of replacing two Gaussians with a single Gaussian. Define the *Gaussian quadratic loss* $\mathrm{GQL}(p)$ as the squared difference between a distribution p and the Gaussian distribution with the same mean m and variance v as p:

$$\mathrm{GQL}(p) = \mathrm{SD}(p \| \mathcal{N}(m, v)). \tag{20}$$

Lemma 3: For the distribution which is the mixture of two Gaussians $t_1 = (m_1, v_1, c_1)$ and $t_2 = (m_2, v_2, c_2)$ with $c_1 + c_2 = 1$, the Gaussian quadratic loss is given by:

$$\begin{aligned} &\mathrm{GQL}(t_1, t_2) = \\ &\quad \frac{1}{2\sqrt{\pi v}} + \frac{c_1^2}{2\sqrt{\pi v_1}} + \frac{c_2^2}{2\sqrt{\pi v_2}} \\ &\quad - \frac{2c_1}{\sqrt{2\pi(v + v_1)}} e^{-\frac{(m-m_1)^2}{2(v+v_1)}} - \frac{2c_2}{\sqrt{2\pi(v + v_2)}} e^{-\frac{(m-m_2)^2}{2(v+v_2)}} \\ &\quad + \frac{2c_1c_2}{\sqrt{2\pi(v_1 + v_2)}} e^{-\frac{(m_1-m_2)^2}{2(v_1+v_2)}}, \end{aligned} \tag{21}$$

where m and v are given in (14) and (15), respectively.

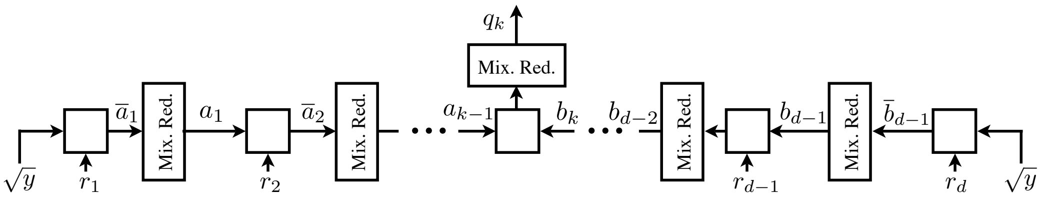

Fig. 1. Forward-backward variable node decoding, with Gaussian mixture reduction applied after each step.

Even when the mixture is not normalized, the GQL is computed by first normalizing the mixing coefficients. Let $\bar{t}_1 = (m_1, v_1, \bar{c}_1)$ and $\bar{t}_2 = (m_2, v_2, \bar{c}_2)$ with $\bar{c}_1 + \bar{c}_2 \neq 1$. Then $c_i = \bar{c}_i/(\bar{c}_1 + \bar{c}_2), i = 1, 2$, and $\text{GQL}(\bar{t}_1, \bar{t}_2)$ is defined to be equal to $\text{GQL}(t_1, t_2)$ as found using Lemma 3.

B. Approximating N Gaussians with M Gaussians

This section describes the Gaussian mixture reduction algorithm, which combines Gaussians in a greedy pair-wise fashion until a stopping condition is reached. Of the Gaussians remaining at each step, the two with the lowest GQL distance metric are replaced by the single Gaussian with the same moments, resulting in one fewer Gaussians. This process is repeated iteratively until a stopping condition is reached.

The algorithm input is a mixture of N Gaussians, $f(z)$, as defined in (6), given as a list of triples. The algorithm output is a list of M triples of means, variances and mixing coefficients, $\{\ (m_1^{\mathsf{m}}, v_1^{\mathsf{m}}, c_1^{\mathsf{m}}),\ \ldots,\ (m_M^{\mathsf{m}}, v_M^{\mathsf{m}}, c_M^{\mathsf{m}})\ \}$ with $\sum_{i=1}^{M} c_i^{\mathsf{m}} = 1$, that also forms a Gaussian mixture $g(z)$. With $M \leq N$, the output mixture should be a good approximation of the input mixture, $g(z) \approx f(z)$. The algorithm is denoted as $g(z) = \text{GMR}(f(z))$.

Two conditions must be satisfied for the algorithm to stop. First, the GQL gives the error incurred as a result of combining at each step, and combining continues while the GQL is greater than a specified threshold θ. This local, one-step error is a surrogate for a global error metric. Second, the algorithm continues combining as long as the number of Gaussians is greater than the specified M^{max}.

Gaussian Mixture Reduction Algorithm

1) Input:
   a) List $\mathcal{L} = \{t_1, t_2, \ldots, t_N\}$ of N triples describing a Gaussian mixture.
   b) Two stopping parameters, θ and M^{max}.
2) Initialize:
   a) The current search list, $\mathcal{C}$, is the input list: $\mathcal{C} \leftarrow \mathcal{L}$.
   b) The length of current list, $M^{\mathsf{c}} \leftarrow N$.
   c) The current error, θ^{c} is the minimum GQL between all pairs of Gaussians:
   $$\theta^{\mathsf{c}} \leftarrow \min_{t_i, t_j \in \mathcal{C}, i \neq j} \text{GQL}(t_i, t_j).$$
3) Greedy combining. While $\theta^{\mathsf{c}} < \theta$ or $M^{\mathsf{c}} > M^{\mathsf{max}}$:
   a) Determine the pair of Gaussians (t_i, t_j) with the smallest GQL:
   $$(t_i, t_j) \quad \leftarrow \quad \arg \min_{t_i, t_j \in \mathcal{C}, i \neq j} \text{GQL}(t_i, t_j).$$
   b) Add the single Gaussian with the same moment as t_i and t_j to the list:
   $$\mathcal{C} \leftarrow \mathcal{C} \cup \text{MM}(t_i, t_j).$$
   c) Delete t_i and t_j from list: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{t_i, t_j\}$.
   d) Decrement the current list length: $M^{\mathsf{c}} \leftarrow M^{\mathsf{c}} - 1$.
   e) Recalculate the minimum GQL:
   $$\theta^{\mathsf{c}} \quad \leftarrow \quad \min_{t_i, t_j \in \mathcal{C}, i \neq j} \text{GQL}(t_i, t_j).$$
4) Output: list of M triples given by $\mathcal{C}$.

It is desirable to repeat step 3 as long as the current reduced Gaussian function $\mathcal{C}$ remains a good approximation of the input function $\mathcal{L}$, however both stopping conditions must be satisfied. The one-step GQL error may be greater than the specified threshold θ if the number of remaining Gaussians is not yet M^{max}. On the other hand, the number of output Gaussians may be less than M^{max}, if the one-step error is sufficiently low.

At each step, the new Gaussian minimizes the divergence with the two it replaced, which is locally optimal. However, by Lemma 2 since the squared distance metric is only a lower bound on the divergence, it is not guaranteed that the pair with the lowest divergence will be selected. Further, this is a greedy algorithm, and no claims are made about its global optimality.

IV. LDLC Decoding Using Gaussian Mixture Reduction

A. LDLC Decoder

The Gaussian mixture reduction algorithm is applied to the belief-propagation decoder to stop the exponential growth of the number of Gaussians in each mixture. In order to apply the algorithm efficiently, the following refinement is necessary.

For any LDLC decoder, the complexity at both the check node and variable node can be reduced by decomposing the computations using a forward-backward-type algorithm. The outputs are found by combining recusively-computed forward messages $a(z)$ and backward messages $b(z)$.

For the Gaussian-mixture LDLC decoder in particular, the Gaussian mixture reduction algorithm is applied after each forward and backward recursion step, and is illustrated for the variable node in Fig. 1. The forward messages at step k are $a_k(z)$ and $\bar{a}_k(z)$, before and after mixture reduction, respectively. The forward recursion is initialized as:

$$a_0(z) \quad = \quad \sqrt{y(z)}. \tag{22}$$

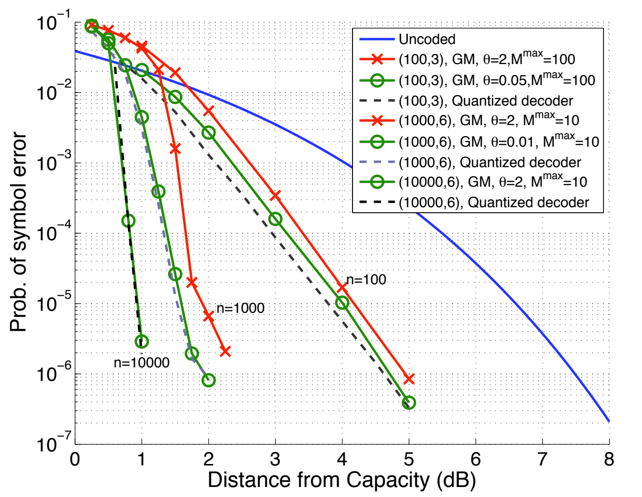

Fig. 2. Symbol error rate for gaussian mixture (GM) and quantized decoder.

Two operations are performed at each recursion step. First, compute the message $\overline{a}_k(z)$. Then, the Gaussian mixture reduction algorithm is applied. For $k = 1, 2 \ldots, d$,

$$\overline{a}_k(z) = a_{k-1}(z) \cdot r_k(z), \text{ and} \tag{23}$$

$$a_k(z) = \text{GMR}(\overline{a}_k(z)). \tag{24}$$

The backward messages $b_k(z), \overline{b}_k(z)$ are found similarly, including initialization $b_d(z) = \sqrt{y(z)}$. When $y(z)$ is a single Gaussian, $\sqrt{y(z)}$ is also Gaussian with the same mean and twice the variance. By initializing in this way, many of the periodic elements of $r_1(z)$ and $r_d(z)$, after multiplication by $\sqrt{y(z)}$, have very small mixing coefficients, which are combined by the Gaussian mixture reduction algorithm. This effect persists as the forward-backward recursions proceed, and in both directions. The output message $r_k(z)$ is the product $a_{k-1}(z) \cdot b_k(z)$, and so the output message properly includes $(\sqrt{y(z)})^2$.

At the check node, a similar forward-backward decomposition can be performed, and the Gaussian mixture reduction algorithm is applied after each recursion step. Gaussian mixture reduction is applied to the summand $\mathcal{N}(z; m_d(b), v_d)$ in (7), but not to the periodic message $r_k(z)$.

As iterations progress, the message variances decrease towards a single narrow Gaussian. But due to quantization effects, two peaks may not align, and the variable-node product will have a large error, leading to instabilities. This was avoided by setting a minimum variance for a mixture. A message variance v_i is replaced by $\max(v_i, \gamma \cdot \sigma^2)$, where γ is a constant, perhaps 0.05, and σ^2 is the channel noise variance, so the minimum value scales with the channel noise. The quantized implementation uses a distinct method to avoid instabilities [1].

Simulation results comparing the proposed decoder with the quantized decoder are shown in Fig. 2. For the dimension 100 lattice, the performance loss was no greater than 0.2 dB. For the dimension 1000 and 10000 lattice, there was no discernible performance loss at high signal-to-noise ratios. One-step error threshold $\theta = 0.05, 2$ were considered.

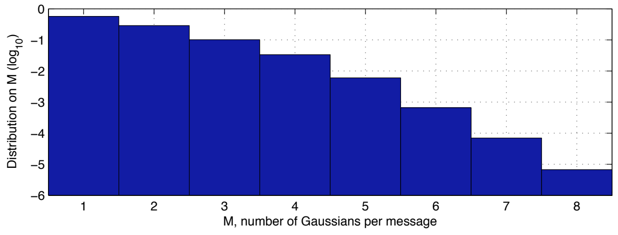

Fig. 3. Distribution of M for $q_k(z)$, decoding a $(1000, 6)$ lattice, $\theta = 0.05$, distance to capacity 2 dB, iteration number 5.

B. Complexity

The complexity of the decoding algorithm is proportional to M^4, where M is the number of Gaussians in a mixture. In each step of the node decomposition, the two inputs (e.g. a_1, r_2) produce an output (e.g. $\overline{a}_2$) consisting of M^2 Gaussians. In the Gaussian mixture reduction algorithm, the primary complexity is computing the initial GQL between k pairs, requiring k^2 operations. With $k = M^2$, we obtain an overall complexity proportional to M^4. In an investigation of the $(1000, 6)$ code, it was observed that the average number of Gaussians decreased for increasing signal-to-noise ratio and increasing one-step error θ. When $\theta = 0.05$ values of M greater than 8 were observed, but rarely (see Fig. 3), and for $\theta = 2$, values larger than 3 were never observed. Single Gaussian messages were always most common. Thus, while the complexity is proportional to M^4, the distribution of M depends indirectly on other parameters.

The complexity of the quantized algorithm is dominated by a discrete Fourier transform of size $1/\Delta$ where Δ is the quantization bin width, $\Delta = 1/128$ was used in the simulations. It is difficult to directly make comparisons of the computational complexity of the two algorithms.

The memory required for the Gaussian mixture reduction algorithm, however, is significantly superior to the quantized algorithm. The proposed algorithm requires storage of $3M$ (for the mean, variance and mixing coefficient), for each message. The quantized algorithm, however used 1024 quantization points for each message.

References

[1] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Transactions on Information Theory*, vol. 54, pp. 1561–1585, April 2008.
[2] D. W. Scott and W. F. Szewczyk, "From kernels to mixtures," *Technometrics*, vol. 43, pp. 323–335, August 2001.
[3] S. Sarvotham, D. Baron, and R. G. Baraniuk, "Compressed sensing reconstruction via belief propagation," Tech. Rep. ECE-06-01, Department of Electrical and Computer Engineering, Rice University, July 2006.
[4] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Transactions on Information Theory*, vol. 40, pp. 409–417, March 1994.
[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.