Iterative Encoding with Gauss-Seidel Method for Spatially-Coupled Low-Density Lattice Codes

Hironori Uchikawa*, Brian M. Kurkoski†, Kenta Kasai* and Kohichi Sakaniwa*

*Dept. of Communications and Integrated Systems, Tokyo Institute of Technology, 152-8550 Tokyo, Japan.
Email: {uchikawa, kenta, sakaniwa}@comm.ss.titech.ac.jp
†School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), 923-1292 Ishikawa, Japan.
Email: kurkoski@jaist.ac.jp

***Abstract*—While it is known that spatially-coupled low-density lattice codes (SC-LDLC) have better decoding performance than conventional (non-coupled) LDLC lattices, in this paper it is shown that their encoding complexity is also lower. Since non-zero elements are mainly in lower triangular entries of the sparse inverse generator matrix of SC-LDLC, iterative encoding with the Gauss-Seidel method performs well. The convergence speed of iterative encoding is evaluated by both the mean square error (MSE) and the symbol error rate between a given integer vector $\underline{b}$ and the inversely generated integer vector from the codeword of $\underline{b}$. Numerical experiments show that the convergence of encoding for SC-LDLC is 3 times faster than that of the conventional LDLC, at an MSE of 10^{-10} for dimension $n = 10000$.**

I. Introduction

The additive white Gaussian noise (AWGN) channel is an important channel both from theoretical and practical points of view. In 1959, Shannon found the capacity of the AWGN channel using random coding [1]. Due to the lack of structure, random codes are not practical as channel codes. Almost 40 years later, it was proved that lattice codes achieve the channel capacity [2], [3]. Although lattice codes have elegant structure, they are not efficiently decodable codes, i.e. their decoding complexity is not linear in the dimension. In 2008, Sommer et al. proposed efficiently decodable lattice codes called low-density lattice codes (LDLC), by defining a sparse inverse generator matrix [4]. Since Sommer et al. did not use shaping methods, the term LDLC lattices is used in this paper. Although LDLC lattices can be decoded efficiently using belief-propagation (BP) algorithm, capacity-achieving LDLC lattices have not so far been constructed. The noise threshold of LDLC lattices in [4] appeared within 0.5 dB of the capacity of the unconstrained-power AWGN channel.

In [5], we developed a new LDLC lattice construction, called spatially-coupled (SC) LDLC lattices, based upon spatial coupling principles [6]. Evaluation was performed using Monte Carlo density evolution using a single-Gaussian approximated BP decoder [7]. While the conventional lattice construction leaves a gap of 0.5 dB to capacity, the SC-LDLC lattice construction reduces this gap to 0.33 dB from capacity[1], of the unconstrained power channel.

Although the decoder's computational complexity is only $O(n)$, generating lattice points requires computational complexity of $O(n^2)$ in general. In [4], Sommer et al. suggested that linear computational complexity encoding can be accomplished using the Jacobi method, which is an iterative algorithm for determining the solution of a system of linear equations. Sommer et al. described that the convergence of the method was guaranteed by the nature of the sparse inverse generator matrix of LDLC lattices. However, they did not investigate convergence speed of such iterative encoding methods. Since processing over many iterations is time- and power-consuming, not only convergence conditions but also convergence speed is important for practical applications. In this paper, we evaluate the convergence speed of encoding using the Gauss-Seidel method, another iterative algorithm [8]. Numerical experiments show that the SC-LDLC encoder with the Gauss-Seidel method has significantly faster convergence speed than the LDLC encoder with the Gauss-Seidel method because of the spatially-coupled structure.

II. LDLC and SC-LDLC Lattices

A. Lattices

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

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

where $\mathbf{b} = (b_1, \ldots, b_n)^\mathsf{T}$ is from the set of all possible integer vectors, $b_i \in \mathbb{Z}$. The transpose of a vector $\mathbf{x}$ is denoted $\mathbf{x}^\mathsf{T}$. 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. It is assumed that $\mathbf{G}$ is n-by-n and full rank (note SC-LDLC lattices allow $\mathbf{G}$ to have additional rows which are linearly dependent).

The decoding performance of lattices are evaluated over the unconstrained-power AWGN channel [4], [5]. 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 codeword symbol. Then the received sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\mathsf{T}$ is

[1]Previously we reported 0.22 dB gap from capacity [5], but the correct value is 0.33 dB.

978-1-4673-2579-0/12/$31.00 ©2012 IEEE

$y_i = x_i + z_i$, for $i = 1, 2, \ldots, n$. The maximum-likelihood decoder selects $\hat{\mathbf{x}} \in \Lambda$ as the estimated codeword, and a decoding error is declared if $\hat{\mathbf{x}} \neq \mathbf{x}$. The capacity of this channel is the maximum noise power at which a maximum-likelihood decoder can recover the transmitted lattice point with error probability as low as desired for sufficiently large block length. In the limit that n becomes asymptotically large, there exist lattices which satisfy this condition if and only if [9]:

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

In the above $|\det(\mathbf{G})|$ is the volume of the Voronoi region, which is inversely proportional to lattice density.

B. LDLC lattices

An LDLC lattice is a lattice which has a sparse inverse generator matrix $\mathbf{H}$. Since the inverse generator matrix $\mathbf{H} = \mathbf{G}^{-1}$ of LDLC lattices is sparse, LDLC lattices can be decoded using BP [4]. An $(n, \alpha,\, d)$ LDLC lattice is defined by the n-by-n matrix $\mathbf{H}$ which has row and column weight d, where each row and column has one entry of weight ± 1 and $d-1$ entries with weight which depends upon α. More precisely, the matrix $\mathbf{H}$ is defined as:

$$\mathbf{H} = \mathbf{P}'_1 + w \sum_{i=2}^{d} \mathbf{P}'_i, \tag{3}$$

where

$$\mathbf{P}'_i = \mathbf{S}_i \mathbf{P}_i. \tag{4}$$

$\mathbf{S}_i$ denotes a random sign change matrix, $\mathbf{P}_i$ denotes a random permutation matrix, and

$$w = \sqrt{\frac{\alpha}{d-1}}.$$

We choose $0 \leq \alpha < 1$, so that BP decoding of LDLC lattices will converge exponentially fast [4]. The permutation matrices are not chosen in a totally random manner but so as to generate $\mathbf{H}$ having exactly one ± 1 and exactly $d-1$ $\pm w$'s in each column and row. The random sign change matrix $\mathbf{S}_i$ is a square, diagonal matrix, where the diagonal entries are $+1$ or -1 with probability 1/2.

C. Spatially-Coupled LDLC lattices

We define an (N, α, d, L) SC-LDLC lattice as a dimension $N(L-d+1)$ lattice with an $NL \times NL$ inverse generator matrix $\mathbf{H}_{[L]}$ as described by Eq. (7). The structure of $\mathbf{H}_{[L]}$ is similar to the parity check matrix of tail-biting convolutional codes. In Eq. (7), $\mathbf{H}^{(l)} = \mathbf{P}'^{(l)}_1 + w \sum_{i=2}^{d} \mathbf{P}'^{(l)}_i$ is an inverse generator matrix of an (N, α, d) LDLC lattice for $l \in \{1, \ldots, L\}$, and each $\mathbf{P}'^{(l)}_i$ represents a distinct matrix of the form of Eq. (4), for distinct l and i.

In this construction, $N(d-1)$ integers are set to 0. The integer vector of form:

$$\tilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{N(d-1)} \end{bmatrix},$$

is used, so that if the (N, α, d, L) SC-LDLC lattice has a lattice point $\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}, \ldots, \mathbf{x}^{(L)})^{\mathsf{T}}$ and $\mathbf{x}^{(l)} = (x_1^{(l)}, \ldots, x_N^{(l)})$, then $\mathbf{H}_{[L]}\mathbf{x} = \tilde{\mathbf{b}}$, where $\mathbf{b} = (\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(l)}, \ldots, \mathbf{b}^{(L-d+1)})^{\mathsf{T}}$ and $\mathbf{b}^{(l)} = (b_1^{(l)}, \ldots, b_N^{(l)})$, is an information integer vector, and, $\mathbf{0}_{N(d-1)}$ is the all zero column vector of length $N(d-1)$. The inverse matrix of $\mathbf{H}_{[L]}$ is defined as $\tilde{\mathbf{G}}_{[L]}$. Since we set $\mathbf{0}_{N(d-1)}$ to the last $d-1$ sections of $\tilde{\mathbf{b}}$, The sub-matrix $\mathbf{G}_{[L]}$ which is from column 1 to $N(L-d+1)$ of $\tilde{\mathbf{G}}_{[L]}$ can be used for generating lattice points. Therefore a lattice point in the dimension $N(L-d+1)$ SC-LDLC lattice is generated as

$$\mathbf{x} = \mathbf{G}_{[L]}\mathbf{b} = \tilde{\mathbf{G}}_{[L]}\tilde{\mathbf{b}}. \tag{5}$$

The dimension of the SC-LDLC lattice is, therefore, less than the number of elements in $\mathbf{x}$, which is $n = NL$. Dimension ratio is defined as

$$R_L = \frac{N(L-d+1)}{NL} = 1 - \frac{d-1}{L}. \tag{6}$$

The ratio R_L converges to 1 with increasing L, with a gap of $O(1/L)$. Therefore, this dimension loss is negligible for sufficiently large L. We observe that the noise threshold of the (N, 0.8, 7, L) SC-LDLC lattices, with sufficiently large L, is very close to the theoretical limit [5].

III. Encoding with the Gauss-Seidel Method

Generally, a lattice encoder generates a lattice point $\mathbf{x}$ using Eq. (1). For LDLC lattices, $\mathbf{G}$ is not sparse in general, and so the encoder requires computational complexity and storage of $O(n^2)$. However, $\mathbf{b} = \mathbf{H}\mathbf{x}$ is a system of linear equations which can be solved using iterative methods. One such method, the Jacobi method, can be described as a "parallel" approach, finding a candidate solution $\mathbf{x}(t)$ on iteration t, using the previous solution $\mathbf{x}(t-1)$. This approach was suggested by Sommer et al., observing that encoding complexity is $O(n)$, since $\mathbf{H}$ is sparse [4].

The Gauss-Seidel method is another iterative method. The Gauss-Seidel method is a "serial" approach; in each iteration, each element of the candidate solution is computed in turn. To perform computations serially, the Gauss-Seidel method decomposes the matrix into upper- and lower-triangular parts. In this section, LDLC and SC-LDLC encoders with the Gauss-Seidel method are described. The Gauss-Seidel method has faster convergence speed than the Jacobi method. Moreover the SC-LDLC encoder with the Gauss-Seidel method performs well because of the spatially-coupled structure.

A. LDLC Encoder with the Gauss-Seidel Method

Before explanation of the Gauss-Seidel Method, we define the matrix $\underline{\mathbf{H}}$ as a row-permuted version of the sparse inverse generator matrix $\mathbf{H}$, such that $\underline{\mathbf{H}}$ has ± 1 in the diagonal entries (non-zero diagonal elements are required by the Gauss-Seidel method). For example, a pictorial view of a matrix $\underline{\mathbf{H}}$ for the (100, 0.8, 7) LDLC lattices is shown in Fig. 1. Also the vector $\underline{\mathbf{b}}$ is a permuted version of the integer vector $\mathbf{b}$, such

$$
\mathbf{H}_{[L]} = \begin{bmatrix}
\mathbf{P}_1'^{(1)} & & & & & w\mathbf{P}_d'^{(L-d+2)} & \cdots & w\mathbf{P}_2'^{(L)} \\
w\mathbf{P}_2'^{(1)} & \mathbf{P}_1'^{(2)} & & & & & \ddots & \vdots \\
\vdots & \vdots & \ddots & & & & & w\mathbf{P}_d'^{(L)} \\
w\mathbf{P}_d'^{(1)} & w\mathbf{P}_{d-1}'^{(2)} & \ddots & \ddots & & & & \\
 & \ddots & \ddots & \ddots & \mathbf{P}_1'^{(L-d+1)} & & & \\
 & & \ddots & \ddots & w\mathbf{P}_2'^{(L-d+1)} & \mathbf{P}_1'^{(L-d+2)} & & \\
 & & & \ddots & \vdots & \vdots & \ddots & \\
 & & & & w\mathbf{P}_d'^{(L-d+1)} & w\mathbf{P}_{d-1}'^{(L-d+2)} & \cdots & \mathbf{P}_1'^{(L)}
\end{bmatrix} \tag{7}
$$

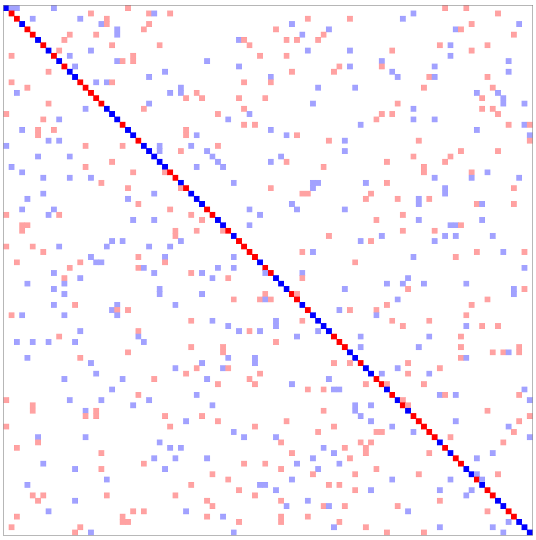

Fig. 1. Pictorial view of a row-permuted matrix $\underline{\mathbf{H}}$ of the $(100, 0.8, 7)$ LDLC lattices. Red (resp. blue) dots represent positive (resp. negative) coefficients. Dark (resp. light) dots represent ± 1 (resp. $\pm w$) coefficients.

that the ith element of $\underline{\mathbf{b}}$ equals the element of $\mathbf{b}$ for which the corresponding row of $\mathbf{H}$ has ± 1 at the ith location.

Using the Gauss-Seidel method [8], an LDLC encoder calculates several iterations of the form:

$$
\mathbf{x}(t+1) = (\underline{\mathbf{L}} + \underline{\mathbf{D}})^{-1}(\underline{\mathbf{b}} - \underline{\mathbf{U}}\mathbf{x}(t)), \tag{8}
$$

where $\underline{\mathbf{D}}$ is the diagonal matrix with the diagonal elements of $\underline{\mathbf{H}}$, and $\underline{\mathbf{L}}$ (resp. $\underline{\mathbf{U}}$) has the lower (resp. upper) triangular elements without the diagonal elements of $\underline{\mathbf{H}}$. Therefore, $\underline{\mathbf{H}}$ is decomposed as:

$$
\underline{\mathbf{H}} = \underline{\mathbf{L}} + \underline{\mathbf{D}} + \underline{\mathbf{U}}.
$$

Denote t as the index of iteration and $\mathbf{x}(0)$ is initialized with $\mathbf{0}_n$. The convergence properties are well-studied, and is related to the spectral radius of the matrix $(\underline{\mathbf{L}}+\underline{\mathbf{D}})^{-1}\underline{\mathbf{U}}$ [8]. However, convergence may still occur even if properties are not satisfied, and arguments concerning convergence are beyond the scope of this paper.

The element-wise equation of Eq. (8) is as follows

$$
\begin{aligned}
x_i(t+1) = \frac{1}{\underline{h}_{i,i}} \Big(\underline{b}_i - & \sum_{j \in \underline{\mathcal{J}}_i \cap \{j | j < i\}} \underline{h}_{i,j} x_j(t+1) \\
& - \sum_{j' \in \underline{\mathcal{J}}_i \cap \{j' | j' > i\}} \underline{h}_{i,j'} x_{j'}(t) \Big),
\end{aligned}
$$

where $x_i(t+1)$ is the ith entry of $\mathbf{x}(t+1)$, $\underline{h}_{i,j}$ is the (i,j) entry of $\underline{\mathbf{H}}$, and $\underline{\mathcal{J}}_i$ is the column index set of the non-zero elements in the row i of $\underline{\mathbf{H}}$. The computation of $x_i(t+1)$ uses the elements of $\mathbf{x}(t+1)$ that have already been computed, and the elements of $\mathbf{x}(t)$ that have not yet been computed at iteration $t+1$.

Consider a sparse inverse generator matrix $\mathbf{H}_{[L]}$ of the SC-LDLC lattices. Non-zero elements are mainly in lower triangular entries of $\mathbf{H}_{[L]}$ without the first $N(d-1)$ rows, see Eq. (7). Since the Gauss-Seidel method uses already-computed elements of $\mathbf{x}(t+1)$ at iteration $t+1$ for $x_i(t+1)$ in the case that the non-zero elements are in $\underline{\mathcal{J}}_i \cap \{j | j < i\}$, it is expected that the convergence speed of the encoding of the SC-LDLC lattices with the Gauss-Seidel method is faster than that of the LDLC lattices.

B. SC-LDLC Encoder with the Gauss-Seidel Method

An SC-LDLC encoder with the Gauss-Seidel method generates a codeword vector $\mathbf{x}^{(l)}(t+1)$ at section l and iteration $t+1$ by Eq. (9). The matrix $\underline{\mathbf{P}}_d'^{(l)}$ is row-permuted version of the $\mathbf{P}_d'^{(l)}$, such that $\underline{\mathbf{H}}_{[L]}$ has ± 1 in the diagonal entries. For example, a pictorial view of a matrix $\underline{\mathbf{H}}_{[L]}$ for a (5, 0.8, 7, 20) SC-LDLC lattice is shown in Fig. 2. The vector $\underline{\mathbf{b}}^{(l)}$ is a permuted version of the integer vector $\mathbf{b}^{(l)}$, such that the ith element of $\underline{\mathbf{b}}^{(l)}$ equals the element of $\mathbf{b}^{(l)}$ for which the corresponding row of $\mathbf{P}_1'^{(l)}$ has ± 1 at the ith location.

All elements of $\mathbf{x}(0)$ are initialized with 0. From Eq. (9), it is observed that the SC-LDLC encoder uses the elements of $\mathbf{x}(t)^{(l)}$ computed at the previous iteration for the first $d-1$ sections. For remaining sections, the SC-LDLC encoder uses only the elements of $\mathbf{x}(t+1)^{(l)}$ computed at the current iteration. Intuitively it is expected that the convergence speed of the encoding of SC-LDLC lattices with the Gauss-Seidel method is faster than that of LDLC lattices.

$$\left(\mathbf{x}^{(l)}(t+1)\right)^{\mathsf{T}} = \begin{cases} (\underline{\mathbf{P}}_1'^{(l)})^{\mathsf{T}} \left((\underline{\mathbf{b}}^{(l)})^{\mathsf{T}} - w \sum_{m=1}^{l-1} \underline{\mathbf{P}}_{m+1}'^{(l-m)} \left(\mathbf{x}^{(l-m)}(t+1)\right)^{\mathsf{T}} - w \sum_{m=l}^{d-1} \underline{\mathbf{P}}_{m+1}'^{(L-m+l)} \left(\mathbf{x}^{(L-m+l)}(t)\right)^{\mathsf{T}} \right) & \text{for } 1 \le l < d, \\ (\underline{\mathbf{P}}_1'^{(l)})^{\mathsf{T}} \left((\underline{\mathbf{b}}^{(l)})^{\mathsf{T}} - w \sum_{m=1}^{d-1} \underline{\mathbf{P}}_{m+1}'^{(l-m)} \left(\mathbf{x}^{(l-m)}(t+1)\right)^{\mathsf{T}} \right) & \text{for } d \le l \le L \end{cases} \tag{9}$$

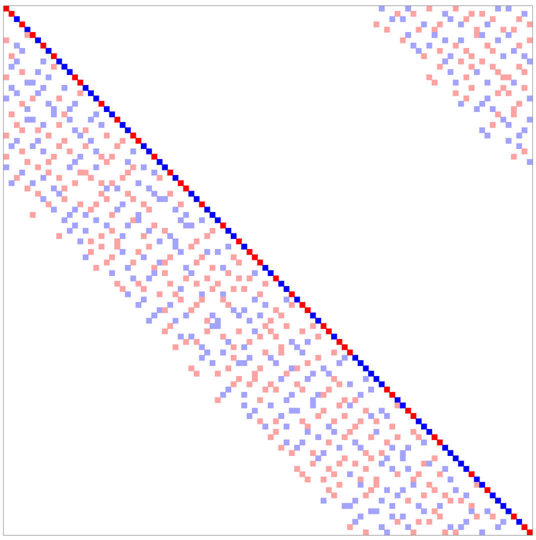

Fig. 2. Pictorial view of a row-permuted matrix $\underline{\mathbf{H}}_{[L]}$ of the $(5, 0.8, 7, 20)$ SC-LDLC lattices. Red (resp. blue) dots represent positive (resp. negative) coefficients. Dark (resp. light) dots represent ± 1 (resp. $\pm w$) coefficients.

Due to the spatially-coupled structure of $\underline{\mathbf{H}}_{[L]}$, the SC-LDLC encoder can be implemented using a transversal filter architecture. The SC-LDLC encoder architecture is shown in Fig. 3. The encoder consists of transversal filter with ring FIFOs. This encoder is simple because the encoder does not need to manage routing connections with all the elements of $\mathbf{x}$ for computations differently from the LDLC encoder.

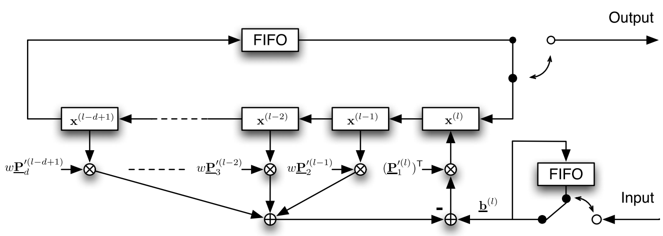

Fig. 3. Architecture of the iterative encoder of SC-LDLC lattices. Note that section index (l) points at $(l + L)$ if the (l) is negative.

IV. Simulation Results

Convergence speed of the encoding with the Gauss-Seidel method is evaluated using Monte-Carlo simulations.

We generate 1000 sparse inverse generator matrices for each lattice ensemble and evaluate 100 randomly generated integer vectors[2] for each matrix to compute the average. Mean square error (MSE) at iteration t is calculated by

$$\mathrm{MSE}(t) = \frac{\left(\underline{\mathbf{b}} - \underline{\mathbf{H}}\mathbf{x}(t)\right)^{\mathsf{T}} \left(\underline{\mathbf{b}} - \underline{\mathbf{H}}\mathbf{x}(t)\right)}{n},$$

where n is the length of $\underline{\mathbf{b}}$. Fig. 4 shows MSE(t) for (1000, 0.8, 7) and (10000, 0.8, 7) LDLC lattices, and (50, 0.8, 7, 20) and (500, 0.8, 7, 20) SC-LDLC lattices. (1000, 0.8, 7) LDLC and (50, 0.8, 7, 20) SC-LDLC $\big($ resp. (10000, 0.8, 7) LDLC and (500, 0.8, 7, 20) SC-LDLC $\big)$ lattices are the same dimension $n = 1000$ (resp. $n = 10000$). It is observed that convergence speed becomes faster with increasing dimension n. Convergence of the (500, 0.8, 7, 20) SC-LDLC lattices is almost 3 times faster than that of the (10000, 0.8, 7) LDLC lattices at an MSE of 10^{-10}. It is also observed that the slopes of the MSE(t) curves for the dimension $n = 1000$ both LDLC and SC-LDLC lattices change around an MSE of 10^{-5}. We conjecture that this is caused by small cycles in the $\underline{\mathbf{H}}$, similar to error floors of low-density parity-check codes. Section size N does not seem to affect the convergence of SC-LDLC lattices because there are small differences in the MSE(t) between the (50, 0.8, 7, 20) and (500, 0.8, 7, 20) SC-LDLC lattices. Fig. 5 shows MSE(t) for the (50, 0.8, 7, L) SC-LDLC lattices for $L = 20, 100, 200$ and 500. We observe that convergence speed accelerates with increasing coupling factor L. From the results in Fig. 5, we can obtain sufficiently accurate codewords for transmission with few iterations if L is sufficiently large.

We also show symbol error rate (SER) performance in Fig. 6. The SER at iteration t is calculated by

$$\mathrm{SER}(t) = \frac{\sum_{i=1}^{n} I\left[\underline{b}_i \neq (\underline{\mathbf{H}}\mathbf{x}(t))_i\right]}{n},$$

where $I[\cdot]$ is an indicator function that returns 1 if an argument is true, otherwise returns 0, and $(\cdot)_i$ denotes ith element of an argument vector. The dimension of lattices in Fig. 6 is 1000, therefore 8 iterations is sufficient for SC-LDLC lattices with the Gauss-Seidel method to vanish encoding symbol error from expectations. On the other hand, 21 iterations is necessary for LDLC lattices. In addition, the Gauss-Seidel method converges slightly faster than Jacobi method for conventional LDLC lattices, and significantly faster for SC-LDLC lattices.

[2] In the simulations, b_i is uniformly distributed over $\{-10, \cdots, 10\}$ for $i \in \{1, \cdots, n\}$.

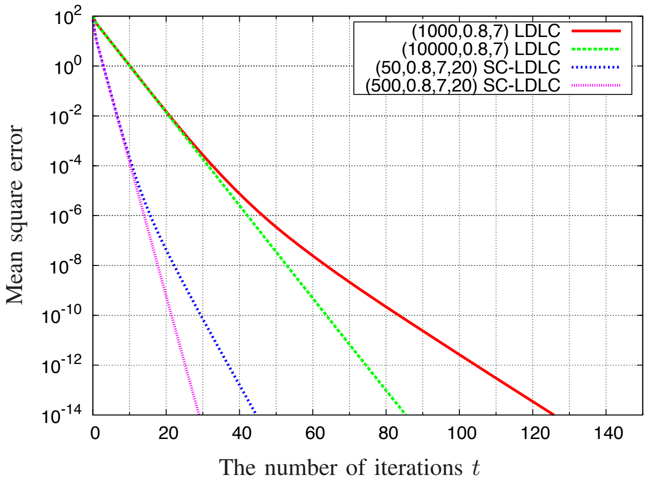

Fig. 4. The number of iterations t versus mean square error of $\underline{\mathbf{b}}$. (1000, 0.8, 7) LDLC and (50, 0.8, 7, 20) SC-LDLC $\big($ resp. (10000, 0.8, 7) LDLC and (500, 0.8, 7, 20) SC-LDLC $\big)$ lattices are the same dimension $n = 1000$ (resp. $n = 10000$). The convergence of SC-LDLC lattices is much faster than that of LDLC lattices at each dimension.

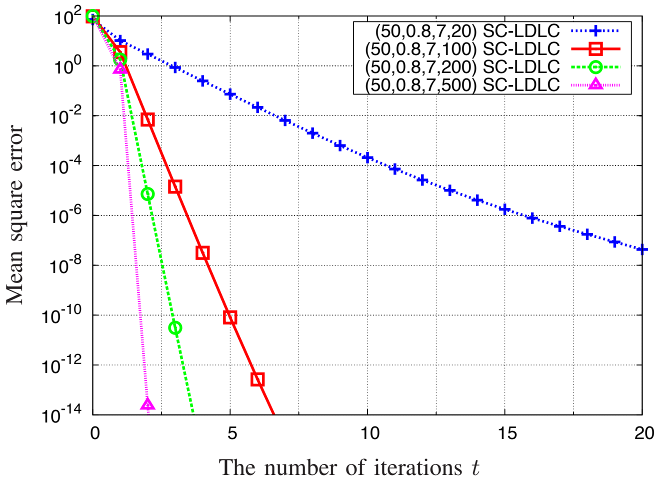

Fig. 5. The number of iterations versus mean square error of $\underline{\mathbf{b}}$. We observe that the convergence speed of SC-LDLC lattices accelerates with increasing coupling factor L.

V. Discussion

We evaluated the convergence performance of iterative encoding with the Gauss-Seidel method for LDLC and SC-LDLC lattices. Numerical experiments showed that the convergence of SC-LDLC lattices is significantly faster than that of LDLC lattices because of the spatially coupled structure.

The Gauss-Seidel method separates the matrix into lower and upper triangular parts. The sparse inverse generator matrix of SC-LDLC lattices is dominated by the lower-triangular components, and since the upper-triangular part is smaller, this may help explain the higher convergence rates for the SC-LDLC lattices.

Note that if SC-LDLC lattices existed where $\mathbf{H}$ is *purely* lower-triangular, then encoding would be particularly straightforward and could be implemented by a simple forward-

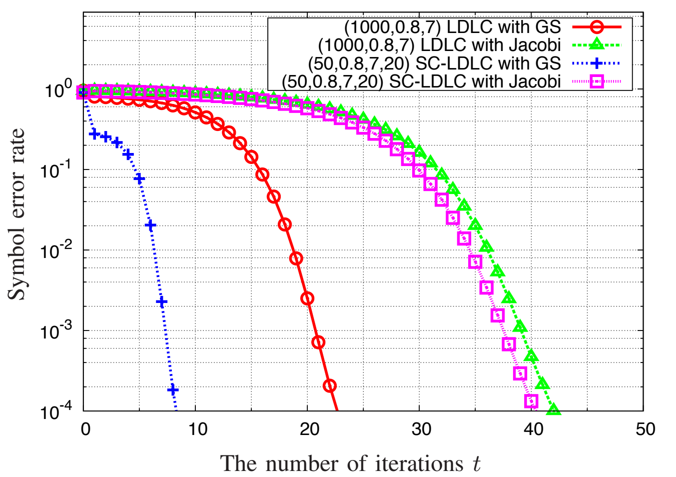

Fig. 6. The number of iterations versus symbol error rate of $\underline{\mathbf{b}}$. The Gauss-Seidel method (GS) and the Jacobi method (Jacobi) are evaluated for both (1000, 0.8, 7) LDLC and (50, 0.8, 7, 20) SC-LDLC lattices. The Gauss-Seidel method converges slightly faster than Jacobi method for conventional LDLC lattices, and significantly faster for SC-LDLC lattices.

substitution. As in many previous papers, shaping was not considered here because of the unconstrained power scenario (i.e. Poltyrev setting). It should be further noted that if a purely lower-triangular matrix is available, then a power constraint can be easily introduced by using cubic shaping, see [10].

References

[1] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell System Technical Journal*, vol. 38, pp. 611–656, May 1959.

[2] R. Urbanke and B. Rimoldi, "Lattice codes can achieve capacity on the AWGN channel," *IEEE Transactions on Information Theory*, vol. 44, pp. 273–278, Jan. 1998.

[3] U. Erez and R. Zamir, "Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, pp. 2293–2314, Oct. 2004.

[4] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Transactions on Information Theory*, vol. 54, pp. 1561–1585, Apr. 2008.

[5] H. Uchikawa, B. M. Kurkoski, K. Kasai, and K. Sakaniwa, "Threshold improvement of low-density lattice codes via spatial coupling," in *Proc. 2012 Int. Conf. on Computing, Networking and Communications (ICNC)*, pp. 1036–1040, Jan. 2012. Hawaii, USA.

[6] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, pp. 803–834, Feb. 2011.

[7] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Single-Gaussian messages and noise thresholds for decoding low-density lattice codes," in *Proc. 2009 IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 734–738, July 2009.

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems*. New York: Society for Industrial and Applied Mathematic (SIAM), 2nd ed., 2003.

[9] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Transactions on Information Theory*, vol. 40, pp. 409–417, Mar. 1994.

[10] N. Sommer, M. Feder, and O. Shalvi, "Shaping methods for low-denisty lattice codes," in *Proc. 2009 IEEE Information Theory Workshop (ITW)*, pp. 238–242, Oct. 2009.