Shaping Methods for Low-Density Lattice Codes

Naftali Sommer, Meir Feder and Ofir Shalvi
Department of Electrical Engineering - Systems
Tel-Aviv University, Tel-Aviv, Israel
Email: meir@eng.tau.ac.il

***Abstract*—Low density lattice codes (LDLC) are recently-proposed lattice codes that can be decoded efficiently and approach the capacity of the additive white Gaussian noise (AWGN) channel. In LDLC a codeword $\underline{x}$ is generated directly at the n-dimensional Euclidean space as a linear transformation of a corresponding integer message vector $\underline{b}$, i.e., $\underline{x} = \boldsymbol{G}\underline{b}$, where $\boldsymbol{H} = \boldsymbol{G}^{-1}$ is restricted to be sparse. In order to design practical lattice codes, the infinite lattice should be combined with a shaping algorithm, that maps information bits to lattice points and ensures that the power of the lattice codewords is properly constrained. This work proposes several efficient and practical shaping algorithms for LDLC.**

I. Introduction

An n dimensional lattice in $\mathbb{R}^m$ is defined as the set of all linear combinations of a given basis of n linearly independent vectors in $\mathbb{R}^m$ with integer coefficients. The matrix $\boldsymbol{G}$, whose columns are the basis vectors, is called a generator matrix of the lattice. Every lattice point is therefore of the form $\underline{x} = \boldsymbol{G}\underline{b}$, where $\underline{b}$ is an n-dimensional vector of integers. From now on, we shall assume that $n = m$ and that the lattice generator matrix is square, but the results can be easily extended to the non-square case.

A lattice code of dimension n is defined by a (possibly shifted) lattice $\boldsymbol{G}$ in $\mathbb{R}^m$ and a shaping region $B \subset \mathbb{R}^m$, where the codewords are all the lattice points that lie within the shaping region B.

Lattice codes can be regarded as the Euclidean space analogue of linear binary codes. It is known for a long time that lattice codes can achieve the capacity of the AWGN channel [1]. Also, in lattice codes both the encoder and the channel use the same real algebra which is natural for the continuous-valued AWGN channel. However, practical coding schemes for the AWGN channel are commonly based on finite alphabet codes, since practical lattice codes with efficient encoding and decoding schemes are not yet available.

Recently, LDLC were proposed [2]. In LDLC, a codeword $\underline{x}$ is generated directly at the n-dimensional Euclidean space as a linear transformation of a corresponding integer message vector $\underline{b}$, i.e., $\underline{x} = \boldsymbol{G}\underline{b}$, where the parity check matrix of the lattice code, defined as $\boldsymbol{H} = \boldsymbol{G}^{-1}$, is restricted to be sparse. The fact that $\boldsymbol{H}$ is sparse was utilized to develop a linear-time iterative decoding scheme which attains, as demonstrated by simulations, good error performance within ~ 0.5dB from capacity at block length of $n = 100,000$ symbols. Furthermore, recently proposed parametric versions of the iterative decoding scheme [3], [4] have low complexity and storage requirements that can compete with any proposed scheme based on finite alphabet codes. Therefore, LDLC have a potential to become a practical and efficient coding scheme for the AWGN channel as well as for Multi-Input, Multi-Output (MIMO) communication systems, where the channel itself generates a lattice.

In [2], methods to construct lattices with good coding gain were discussed, as well as decoding algorithms. Regarding the encoding operation, the LDLC encoder has to calculate $\underline{x} = \boldsymbol{G} \cdot \underline{b}$, where $\underline{b}$ is an integer message vector. Unlike $\boldsymbol{H}$, $\boldsymbol{G} = \boldsymbol{H}^{-1}$ is not sparse, in general, so the calculation requires computational complexity and storage of $O(n^2)$. This is not a desirable property because the decoder's computational complexity is only $o(n)$, so it was suggested in [2] to use the iterative Jacobi method [5] to solve $\boldsymbol{H} \cdot \underline{x} = \underline{b}$, which is a system of sparse linear equations. However, for practical use with the power constrained AWGN channel, the encoding operation must be accompanied by shaping, in order to prevent the transmitted codeword's power from being too large, by making sure that only lattice points that belong to the shaping region B are actually used. This shaping region may be, for example, an n-dimensional sphere. Therefore, instead of mapping the information vector $\underline{b}$ to the lattice point $\underline{x} = \boldsymbol{G} \cdot \underline{b}$, it should be mapped to some other lattice point $\underline{x}' = \boldsymbol{G} \cdot \underline{b}'$, such that the lattice points that are used as codewords belong to the shaping region. The shaping operation is the mapping of the integer vector $\underline{b}$ to the integer vector $\underline{b}'$.

The need for shaping is stronger for LDLC than for conventional coding schemes that are based on binary (or finite-alphabet) codes. For these schemes, the coded bits or symbols are mapped to a finite constellation, so the resulting transmitted signal components are uniformly distributed. Such a signal has a shaping penalty of only $1.53dB$ relative to optimal shaping where the transmitted signal codewords are contained in a hypersphere, and each codeword component has a Gaussian distribution [6]. On the other hand, for LDLC, even if the integer vector $\underline{b}$ is uniformly distributed, the codeword $\underline{x} = \boldsymbol{G} \cdot \underline{b}$ may have very large energy. Therefore, a shaping algorithm is necessary for LDLC, even in order to make the codeword components uniformly distributed, and preferably to achieve the optimal shaping gain.

Practical and efficient shaping methods for LDLC are not yet available. In [2], the simulations were performed with the infinite lattice (no shaping region), and results were compared to the generalized capacity of the the AWGN channel without restrictions, which was defined in [7] as the maximal possible codeword density that can be recovered reliably. A lattice

978-1-4244-4983-5/09/$25.00 © 2009 IEEE

that achieves the generalized capacity of the AWGN channel without restrictions, also achieves the channel capacity of the power constrained AWGN channel, with a properly chosen spherical shaping region [1]. A shaping algorithm for LDLC, based on the iterative LDLC decoding algorithm, was suggested in [8], and was demonstrated for small dimensions. In this paper, several shaping methods for LDLC are proposed. The methods are demonstrated by simulations to give good shaping gains for practical dimensions.

II. Using a Lower Triangular Parity Check Matrix

In [2], Latin square LDLC were defined. In a Latin square LDLC, every row and column of the parity check matrix $\boldsymbol{H}$ has the same d nonzero values, except for a possible change of order and random signs. The sorted sequence of these d values $h_1 \geq h_2 \geq ... \geq h_d > 0$ is referred to as the generating sequence of the Latin square LDLC.

We would like to use a simpler structure for $\boldsymbol{H}$, which will be more convenient for encoding and shaping. First, it is assumed that all the values on the main diagonal of $\boldsymbol{H}$ are 1. This can be assumed, without loss of generality, for any Latin square LDLC that one of its generating sequence elements is 1, by permuting rows and columns of $\boldsymbol{H}$ and by multiplying whole rows by -1 as necessary. Also, we would like $\boldsymbol{H}$ to have a lower triangular structure. Obviously, a parity check matrix of an LDLC can not be lower triangular, since the upper rows (as well as the rightmost columns) can not contain d nonzero elements. Therefore, it is suggested that the column degree of the rightmost column of $\boldsymbol{H}$ will start from 1 and gradually increase until d. In the same manner, the row degree of the top row will be 1, and it will gradually increase until d.

The following matrix is an example of such a matrix with dimension $n = 8$, degree $d = 3$ and generating sequence $(1, 0.7, 0.5)$. The two rightmost columns have a single nonzero element, the next two have 2 nonzero element and the 4 leftmost columns have the final degree $d = 3$. The same is true for the matrix rows, starting from the top.

$$\begin{pmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.7 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.7 & 1.0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0 & 0.7 & 1.0 & 0 & 0 & 0 \\ 0 & -0.7 & 0 & 0.5 & 0 & 1.0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & 0.7 & 0 & 1.0 & 0 \\ 0 & 0 & -0.5 & 0 & 0 & 0.7 & 0 & 1.0 \end{pmatrix}$$

Such a lower triangular matrix can be generated with similar methods to those described in [2] for Latin square LDLC, where the location of each element of the generating sequence in each row can be described by a permutation (since each element appears once, and only once, in each row and column). The difference is that here, instead of a permutation, we shall use a mapping from a subset of the rows (starting at the first row where this element appears, ending at the bottom row) to a subset of the columns (starting at the leftmost column, ending at the rightmost column where this element still appears).

As explained in the next section, the lower triangular structure is very convenient for encoding and shaping. However, it has a drawback: the codeword components whose respective $\boldsymbol{H}$ columns have low degree are less protected. For example, an element whose column has a single nonzero element is effectively uncoded, since it only takes place in a single check equation. As a result, the information integers whose check equations involve less protected codeword components should contain a smaller amount of information, i.e. belong to a smaller constellation. For example, we can use the following constellations for the integers in the example matrix above. The first 2 integers can assume one of 2 possible integer values (i.e. contain 1 bit of information). The next 2 integers can assume one of 4 integer values and contain 2 bits of information, where all the other integers can assume one of 8 possible values and contain 3 bits of information. As shown in Section IV, the rate loss due to this selective constellation reduction can be made relatively small, especially for large codeword length n.

We note that for a lower triangular $\boldsymbol{H}$, the LDLC shaping problem becomes similar to shaping of signal codes [9], [10]. Signal codes are lattice codes for which encoding is done by convolving the information integers with a fixed, finite-length filter. The resulting lattice generator matrix has a Toeplitz structure, which is close to being lower-triangular.

III. Shaping Methods for LDLC

For the methods in Sections III-A-III-C below, it is assumed that $\boldsymbol{H}$ is lower triangular with ones on the diagonal, as defined in Section II. In Section III-D, the methods of Sections III-A and III-C are extended to an arbitrary $\boldsymbol{H}$. We shall assume that each information integer b_i is drawn from the finite constellation $\{0, 1, ..., L-1\}$, where L is the constellation size. As discussed above, we may want to use a different constellation size for each integer, so the constellation size of the i^{th} integer b_i will be denoted by L_i.

A. Hypercube Shaping

Hypercube shaping finds $\underline{b}'$ such that the components of $\underline{x}' = \boldsymbol{G}\underline{b}'$ are uniformly distributed. To achieve that, we shall assume that

$$b'_i = b_i - L_i k_i, \tag{1}$$

where k_i is an integer. The method starts from the first (top) check equation, i.e., from $i = 1$, and continues to $i = n$. For each equation, the value of k_i is chosen such that $|x'_i| \leq L_i/2$, where x'_i is the resulting codeword element, i.e.

$$k_i = \left\lfloor \frac{1}{L_i} \left(b_i - \sum_{l=1}^{i-1} H_{i,l} x'_l \right) \right\rceil \tag{2}$$

The modified integer b'_i is then calculated according to (1), and the codeword element x'_i is then easily calculated as:

$$x'_i = b'_i - \sum_{l=1}^{i-1} H_{i,l} x'_l \tag{3}$$

Since $\boldsymbol{H}$ is sparse, the overall computational complexity is $O(nd)$. At the decoder, the information integers b_i are recovered from b'_i by a simple modulo L_i operation: $b_i = b'_i \mod L_i$.

This method can be interpreted as a generalization of the Tomlinson-Harashima precoding scheme for Inter-Symbol Interference (ISI) channels [11], [12]. Here, the ISI is the contribution of the x'_i components that were already calculated. Therefore, except for singular cases of matrix coefficients, the codeword components will be uniformly distributed.

B. Systematic Shaping

A systematic binary code is defined as a code where the information bits are part of the codeword components. For such codes, the information bits can be easily extracted from the decoded codeword. Also, the information can be easily extracted from an error-free codeword, by simply taking the corresponding systematic bits. This definition can be extended to lattice codes. We shall define a systematic lattice code as a lattice code for which the information integers can be extracted from a noiseless codeword by rounding the codeword components (or some of them, in case of a non-square generator matrix): $b_i = \lfloor x'_i \rceil$.

In order for the shaping and encoding to generate a systematic lattice code, we shall assume that

$$b'_i = b_i - k_i, \tag{4}$$

where k_i is an integer. The method starts from the first (top) check equation. For each equation, the value of k_i is chosen such that $|x'_i - b_i| \leq \frac{1}{2}$, where x'_i is the resulting codeword element, i.e.

$$k_i = -\left\lfloor \sum_{l=1}^{i-1} H_{i,l} x'_l \right\rceil \tag{5}$$

The modified integer b'_i is then calculated according to (4), and the codeword element x'_i is then easily calculated according to (3). At the decoder, the LDLC decoding algorithm naturally estimates x'_i, so b_i can be recovered by simple rounding: $b_i = \lfloor x'_i \rceil$. Alternatively, after estimating b'_i, the values of x'_i can be generated by solving a triangular linear system of sparse equations $\boldsymbol{H} \cdot \underline{x}' = \underline{b}'$, and then the values of b_i are obtained by rounding x'_i.

This method can be interpreted as a generalization of the flexible precoding scheme for ISI channels [13]. With systematic shaping, the coded signal equals the uncoded signal plus an additive "dither" signal, that has magnitude less than $\frac{1}{2}$. Surprisingly, such a small dither signal can yield substantial coding gains.

Unlike hypercube shaping, where the coded signal is always mapped to a hypercube, systematic shaping can be combined with standard constellation shaping algorithms, such as trellis shaping [14] or shell mapping [15], such that additional shaping gain of up to 1.53dB can be potentially obtained. This can be done by applying a constellation shaping algorithm to the uncoded sequence b_i prior to LDLC shaping and encoding. The LDLC encoding and systematic shaping do not alter the shaping properties of the input signal significantly, since they are equivalent to adding a small dither.

C. Nested Lattice Shaping

The hypercube shaping algorithm results in a hypercube shaping domain. As discussed above, it is beneficial to use a spherical shaping domain, since additional 1.53dB of shaping gain can be achieved. However, mapping to a hypersphere is complex, and it is desirable to find simple ways to approximate it.

Consider the hypercube shaping operation $b'_i = b_i - Lk_i$ (for simplicity, we shall assume that the constellation size of all integers is equal to L, but the results can be easily extended to the general case). Suppose that instead of setting k_i in a memoryless manner as in (2), we choose a sequence $\{k_i\}$ that minimizes the energy of the resulting codeword $\sum_i |x'_i|^2$. Using vector notations, we have

$$\underline{b}' = \underline{b} - L\underline{k}. \tag{6}$$

Denote the non-shaped lattice point by $\underline{x} = \boldsymbol{G}\underline{b}$. From (6), we then have $\underline{x}' = \boldsymbol{G}\underline{b}' = \underline{x} - L\boldsymbol{G}\underline{k}$. Choosing $\underline{k}$ that minimizes $\|\underline{x}'\|^2$ is essentially finding the nearest lattice point of the scaled lattice $L\boldsymbol{G}$ to the non-shaped lattice point $\underline{x}$. Therefore, the codewords will be uniformly distributed along the Voronoi cell of the coarse lattice $L\boldsymbol{G}$. The resulting shaping scheme is equivalent to nested lattice coding [16], [1], where the shaping domain of a lattice code is chosen as the Voronoi region of a different, "coarse" lattice, usually chosen as a scaled version of the coding lattice.

Finding the nearest lattice point is exactly the operation of the iterative LDLC decoder. Moreover, unlike decoding, for shaping applications it is not critical to find the exact nearest lattice point, as the result will only be a slight penalty in signal power. Unfortunately, simulations show that the iterative decoding finds a vector $\underline{k}$ with poor shaping gain. The reason is that for shaping, the effective noise is much stronger than for decoding, and the iterative decoder fails to find the nearest lattice point (or even an approximation of it) if the noise is too large.

Therefore, alternative algorithms should be considered for finding an approximate nearest lattice point. The triangular structure of $\boldsymbol{H}$ suggests a tree search over all possible sequences $\underline{k}$, starting at the first component and adding the next component in each layer of the tree. Practically, this tree search can be done with simple sub-optimal sequential decoders such as the M-algorithm [17]. This algorithm starts from the first row of $\boldsymbol{H}$, and sequentially goes down along the rows. The input at row i is a list of up to M candidate sequences for $k_1, ..., k_{i-1}$ (where for $i = 1$ the list is initialized with a single empty sequence). Each of the M sequences is extended with all possible values for k_i, and each extended sequence is assigned a score of $\sum_{j=1}^{i} |x'_i|^2$. The scores are sorted, and the M sequences with smallest score are kept as input to the next row. $\underline{k}$ is finally chosen as the sequence with smallest score after processing of the last row $i = n$. The value of M determines both the storage and the computational

complexity of the shaper, which is $O(ndM)$. Note that for an M-algorithm with M=1, nested lattice shaping reduces to Hypercube shaping, where for $M = \infty$, the algorithm is a full exponential tree search that finds the exact solution for $\underline{k}$.

D. Extending the Shaping Methods to a Non-Triangular $\boldsymbol{H}$

The hypercube and nested lattice shaping methods can be extended to an arbitrary LDLC parity check matrix $\boldsymbol{H}$. First, we can decompose $\boldsymbol{H} = \boldsymbol{TQ}$, where $\boldsymbol{T}$ is lower triangular and $\boldsymbol{Q}$ is orthonormal (using QR decomposition of $\boldsymbol{H}^t$). Assuming that b'_i and b_i are related by $\underline{b}' = \underline{b} - L\underline{k}$, we want to find $\underline{k}$ such that $\underline{x}'$ that satisfies $\boldsymbol{H}\underline{x}' = \underline{b}'$ is bounded in a hypercube (for hypercube shaping) or has minimal power (for nested lattice shaping). Substituting $\boldsymbol{H} = \boldsymbol{TQ}$, we get $\boldsymbol{T}\underline{\tilde{x}} = \underline{b}'$, where $\underline{\tilde{x}} = \boldsymbol{Q}\underline{x}'$. Now, $\boldsymbol{T}$ is lower triangular, so we can use the back substitution procedure of (2),(3) in order to calculate $\underline{k}$ such that $\underline{\tilde{x}}$ is bounded in a hypercube. Alternatively, we can use a sequential algorithm such as described in Section III-C in order to calculate $\underline{k}$ such that $\underline{\tilde{x}}$ has minimal power. Then, since $\boldsymbol{Q}$ is orthonormal, the actual LDLC codeword $\underline{x}' = \boldsymbol{Q}^t\underline{\tilde{x}}$ has the same power as $\underline{\tilde{x}}$, i.e. the same shaping gain. For hypercube shaping, $\underline{x}'$ will now belong to a rotated hypercube, and its components will no longer be uniformly distributed, but the overall shaping gain is maintained.

Practically, we do not want to actually handle (and store) the matrix $\boldsymbol{Q}$. Therefore, instead of finding the codeword $\underline{x}'$ by $\underline{x}' = \boldsymbol{Q}^t\underline{\tilde{x}}$, we can use $\underline{k}$ to calculate $\underline{b}'$, and then solve $\boldsymbol{H}\underline{x}' = \underline{b}'$ using methods for solving systems of sparse linear equations, as described in Section I.

The triangular matrix $\boldsymbol{T}$ differs from the triangular matrices that were considered in Sections III-A-III-C in two main aspects. First, the on-diagonal elements of $\boldsymbol{T}$ are not necessarily 1. Therefore, equations (2) and (3) should be modified to $k_i = \left\lfloor \frac{1}{L_i}\left(b_i - \sum_{l=1}^{i-1} T_{i,l}\tilde{x}_l\right)\right\rceil$ and $\tilde{x}_i = \left(b'_i - \sum_{l=1}^{i-1} T_{i,l}\tilde{x}_l\right)/T_{i,i}$, respectively. Also, the constellation size for symbol i should be chosen as $L_i = \lfloor |T_{i,i}L| \rfloor$, where L is the "nominal" constellation size, in order to guarantee that $|\tilde{x}_i| \leq L/2$. Second, $\boldsymbol{T}$ is not necessarily sparse. As a result, the computational complexity and storage requirements of the shaping algorithm are $O(n^2)$. To improve that, we can keep only the J nonzero values of $\boldsymbol{T}$ that have the largest absolute values, and nullify all other elements of $\boldsymbol{T}$. This results in a trade-off between computational complexity and shaping gain: using a larger number of nonzero elements J will enable better shaping gain but at the cost of higher complexity.

IV. Simulation Results

The first three proposed shaping methods were simulated for an LDLC with dimension $n = 10,000$ and degree $d = 7$. The LDLC generating sequence was $\{1, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}, \frac{1}{\sqrt{7}}\}$. The degree of the rows gradually increased from 1 to d, starting at the first row, as described in Table I. The choice of the row where the constellation size increases has been done via some trial-and-error. This point can be further examined to come up with the optimal choice. The degree of the columns is the same as the degree of the rows, starting from the rightmost column and increasing to the left. The table also shows the constellation that was used for each integer. In principle, the constellation can increase one integer at a time instead of one bit at a time, but integer numbers of bits were used for convenience. The shaping algorithm was simulated with the AWGN channel and the LDLC decoder, as described in [2], such that the input to the decoder was $y_i = x'_i + n_i$, where n_i is white Gaussian noise with variance σ^2 and x'_i is the codeword component. The constellation for each integer was shifted such that its mean is zero. SNR was defined as the ratio between the average signal energy $E\left\{x'^2_i\right\}$ and the noise variance σ^2.

TABLE I
Row/Column degree and constellation size for an upper triangular LDLC

rows	degree	constellation size of appropriate integer
1-50	1	2
51-150	2	2
151-250	3	4
251-500	4	4
501-1,000	5	8
1,001-2,000	6	8
2,001-10,000	7	8

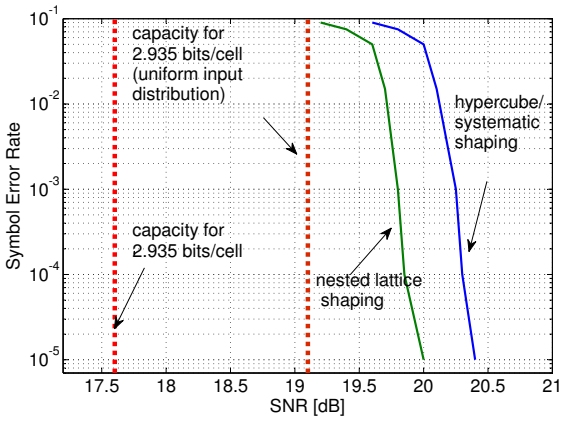

Fig. 1. Simulation results

If we could use a constellation size of 8 for all integers, the information rate was 3 bits/integer. Practically, the information rate will be lower, due to the decrease in constellation size for the less-protected integers, as explained in Section II. The effective rate is then $(150 \cdot 1 + 350 \cdot 2 + 9500 \cdot 3)/10000 = 2.935$ bits/integer, which reflects a loss of 0.065 bits/integer. As noted above, with a more careful choice this loss may be reduced. Channel capacity for 3 bits/cell transmission is at SNR=18dB, where for 2.935 bits/cell it is 17.6dB. Therefore, the information rate loss is equivalent to an SNR loss of 0.4dB. Under uniform channel input distribution constraint, capacity is approximately 1.5dB higher than unconstrained capacity.

Figure 1 shows the symbol error rate (SER) as a function of SNR (where a symbol refers to an information integer). As expected, hypercube shaping and systematic shaping have the same performance, since both generate a signal which is uniformly distributed. These shaping methods can achieve SER of 10^{-5} for SNR of 1.3dB from the channel capacity under uniform distribution constraint (2.8dB away from the unconstrained channel capacity). For ideal shaping, it was

demonstrated in [2] that for $n = 10,000$ the gap to capacity at SER=10^{-5} is 0.8dB. The remaining loss is therefore related to the shaping algorithm. In further study we plan to utilize the option of trellis shaping or shell mapping over the constellation for the systematic shaping scheme, hoping to achieve this way much of the possible 1.53dB shaping gain.

The nested lattice shaping gives better results than the other schemes by 0.4dB. This is rather disappointing, as it could be expected to give a gain of up to the optimal shaping gain of 1.53dB. Moreover, for signal codes, this method indeed gives close to optimal shaping gains [10]. The reason for the smaller gain here may be the fact that in signal codes, $\boldsymbol{G}$ is Toeplitz, so the nonzero elements in a column are concentrated in a small number of consecutive rows. The sequential M-algorithm works well, since the values that are set for k_i only influence several consecutive rows. On the other hand, in LDLC the nonzero elements in a column may be spread along the column. Values that are set for k_i in a certain row, may influence a distant row, but when the M-algorithm reaches the distant row, the correct paths are already thrown away.

Finally, the method of Section III-D was simulated for a non-triangular Latin square $\boldsymbol{H}$ with the same generating sequence as above, using hypercube shaping. Due to computational complexity constraints, the QR decomposition could only be used for dimensions of up to 1000. For dimension 1000, the simulation used nominal data rate of $L = 3$ bits/integer, where the constellation of each integer was set to $L_i = \lfloor |T_{i,i}L| \rfloor$, as explained in Section III-D, resulting in an average data rate of 2.9 bits/integer, for which unconstrained channel capacity is at SNR=17.4dB and the capacity under uniform input constraint is approximately 18.9dB. SER=10^{-5} was achieved at SNR of 20.5dB, 20.6dB, 21.1dB, 22.2dB for J=500000, 200000, 100000, 50000, respectively, where J indicates the number of nonzero elements that were used in the triangular matrix $\boldsymbol{T}$. For J=500000 (which effectively means no truncation at all, since for dimension n=1000 this is approximately the number of nonzero elements in a full triangular matrix) the distance from constrained capacity is 1.6dB. For ideal shaping, it was demonstrated in [2] that for $n = 1000$ the gap to capacity at SER=10^{-5} is 1.5dB, which means that the shaping algorithm loss is only 0.1dB. For J=200000 (average of 200 nonzero elements per row), the algorithm has 0.2dB loss, where for 100 and 50 average nonzeros per row the loss increases to 0.7dB and 1.8dB, respectively. Note that for the shaping algorithms that use a triangular $\boldsymbol{H}$, the number of nonzeros per row is $d = 7$, which results in much smaller complexity and storage.

V. Conclusion

Low density lattice codes provide, for the first time, close to optimal, practical lattice codes that can be encoded and decoded efficiently. Originally, the performance of these codes has been considered due to their structure as infinite lattices. However, for practical application in communication, the lattice codewords should be constrained to a shaping region. This paper describes methods that can efficiently generate power limited lattice points.

The common theme behind these methods is to build the "parity-check" matrix of the lattice (the inverse of its generator matrix) as a triangular matrix, or to convert it to this form using QR decomposition. As demonstrated in this paper, the resulting practical codes can be about 0.8-1.3dB from the uniform prior capacity, which is 1.53dB above the (Gaussian) Shannon capacity. For further study, the paper also discusses techniques with potential to close much of this gap.

One additional interesting point presented in the paper is the notion of "systematic lattice codes" where the integer information symbols can be obtained by rounding the components of the real lattice codeword. We believe that this notion can be useful to shaping (as discussed above) but also for further efficient decoding at high SNR, and may guard the code from error floor phenomena. The study of these topics is left for further research.

Acknowledgment

The authors would like to thank Mr. Yair Yona for discussions on this subject and on efficient LDLC decoding methods, and Prof. Ehud Weinstein for his support and for interesting discussions. The authors also like to thank the anonymous reviewers for their thorough review and valuable comments.

References

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

[2] N. Sommer, M. Feder and O. Shalvi, "Low Density Lattice Codes," *IEEE Trans. Inform. Theory*, vol. 54, pp. 1561-1585, April 2008.

[3] B. M. Kurkoski and J. Dauwels, "Message-Passing Decoding of Lattices Using Gaussian Mixtures," Proceeding of the *Int. Symp. on Inform. Theory, ISIT 2008*, Toronto, Canada, July 2008.

[4] Y. Yona and M. Feder, "Efficient Parametric Decoder of Low Density Lattice Codes", Proceeding of the *Int. Symp. on Inform. Theory, ISIT 2009*, Seoul, Korea, June 2009.

[5] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematic (SIAM), 2nd edition, 2003.

[6] G. D. Forney Jr. and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Trans. Inf. Theory*, pp. 2384–2415, Oct. 1998.

[7] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Trans. Inform. Theory*, vol. 40, pp. 409-417, Mar. 1994.

[8] B. M. Kurkoski, J. Dauwels and H. A. Loeliger, "Power-Constrained Communications Using LDLC Lattices", Proceeding of the *Int. Symp. on Inform. Theory, ISIT 2009*, Seoul, Korea, June 2009.

[9] O. Shalvi, N. Sommer and M. Feder, "Signal Codes," *proceedings of the Information theory Workshop*, 2003, pp. 332–336.

[10] O. Shalvi, N. Sommer and M. Feder, "Signal Codes," submitted to *IEEE Transactions on Information Theory*, available at http://arxiv.org/PS_cache/arxiv/pdf/0806/0806.4773v1.pdf.

[11] M. Tomlinson, "New automatic equalizer employing modulo arithmetic," *Elect. Letters*, pp. 138–139, March 1971.

[12] G. D. Forney and M. V. Eyuboglu, "Combined equalization and coding using precoding," *IEEE Commun. Mag.*, vol. 29, pp. 25-34, Dec. 1991.

[13] R. Laroia, S. A. Tretter and N. Farvardin, "A Simple and Effective Precoding Scheme for Noise Whitening on Intersymbol Interference Channels," *IEEE trans. comm.*, vol. 41, pp. 1460-1463, Oct. 1993.

[14] G. D. Forney Jr., "Trellis Shaping," *IEEE Tran. Inform. Theory*, vol. IT-38(2), pp. 281–300, March 1992.

[15] R. Laroia, N. Farvardin and S. A. Tretter, "On optimal shaping of multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1044-1056, July 1994.

[16] J. H. Conway and N. J. A. Sloane, "A Fast Encoding Method for Lattice Codes and Quantizers," *IEEE Trans. Inf. Theory*, pp. 820–824, Nov. 1983.

[17] T. Aulin, "Breadth-First Maximum Likelihood Sequence Detection: Basics," *IEEE Trans. Comm.*, vol. 47(2), pp. 208–216, Feb 1999.