

# Construction of Irregular LDPC Convolutional Codes with Fast Encoding

Ali Emre Pusane, Kamil Sh. Zigangirov, and Daniel J. Costello, Jr.

Department of Electrical Engineering  
University of Notre Dame  
Notre Dame, IN 46556, U.S.A.

E-mail: {Pusane.1, Zigangirov.1, Costello.2}@nd.edu

**Abstract**—We propose a novel code design technique for irregular LDPC convolutional codes. The constructed codes can be encoded continuously in real time with the help of a shift-register based encoder. For moderate values of the syndrome former memory, simulation results show that the constructed codes outperform LDPC block codes with comparable hardware (processor) complexity.

## I. INTRODUCTION

In the last decade, the coding community has experienced the arrival of the capacity-approaching era, initiated by the invention of turbo coding in 1993 [1]. A few years after the invention of turbo codes, researchers became aware that Gallager's low-density parity-check (LDPC) block codes, first introduced in [2], were also capable of capacity-approaching performance on a variety of channels. Analysis and design of these codes quickly attracted considerable attention in the literature, beginning with the work of Wiberg [3], MacKay and Neal [4], and many others. The irregular version of these codes was first introduced by Luby *et al.* in [5], [6], and analytical tools were presented in [7], [8] to obtain performance limits when decoded using the iterative message-passing algorithms suggested by Tanner [9]. These tools have been successfully employed to design irregular codes that achieve near capacity performance [10], [11]. The convolutional counterparts of LDPC block codes, namely LDPC convolutional codes, were subsequently proposed in [12]. Analogous to LDPC block codes, LDPC convolutional codes are defined by sparse parity-check matrices, which allow them to be decoded using an iterative message-passing algorithm.

Recent studies have shown that these codes are suitable for practical implementation in a number of different communication scenarios, including continuous transmission as well as block transmission in frames of arbitrary size [13], [14], [15]. LDPC convolutional codes are also known for their encoding simplicity. The

original construction proposed in [12] presents a shift-register based systematic encoder for real time encoding of continuous data. This is an advantage when compared to randomly constructed LDPC block codes. Irregular versions of LDPC convolutional codes, which have not been previously considered in the literature, are the subject of this paper. The original code construction technique presented in [12], which generates regular LDPC convolutional codes by tiling permutation matrices of row/column weight 1, does not apply to irregular degree distributions.

We propose a novel code construction technique for irregular LDPC convolutional codes. We do this while preserving the fast encoding property for the constructed codes, i.e., the constructed codes can be encoded continuously in real time with a shift-register based encoder.

The paper is organized as follows. In Section II, we give a brief introduction to LDPC convolutional codes. Following our discussion of the fast encoding property for these codes in Section III, we present a novel way to construct irregular LDPC codes in Section IV. Finally, simulation results are given in Section V, where we compare the bit error rate (BER) performance of moderate sized LDPC convolutional codes with corresponding LDPC block codes, and some conclusions are drawn in Section VI.

## II. LDPC CONVOLUTIONAL CODES

An  $(m_s, J, K)$  regular time-varying LDPC convolutional code is the set of sequences  $\mathbf{v}$  satisfying the equation  $\mathbf{v}\mathbf{H}^T = 0$ , where

$$\mathbf{H}^T = \begin{bmatrix} \mathbf{H}_0^T(0) & \cdots & \mathbf{H}_{m_s}^T(m_s) & & \\ & \ddots & & \ddots & \\ & & \mathbf{H}_0^T(t) & \cdots & \mathbf{H}_{m_s}^T(t+m_s) \\ & & & \ddots & \\ & & & & \ddots \end{bmatrix}. \quad (1)$$

Here,  $\mathbf{H}^T$  is the semi-infinite syndrome former (transposed parity-check) matrix. For a rate  $R = b/c$ ,  $b < c$ , LDPC convolutional code, the elements  $\mathbf{H}_i^T(t)$ ,  $i = 0, 1, \dots, m_s$ , are binary  $c \times (c - b)$  submatrices defined as

$$\mathbf{H}_i^T(t) = \begin{bmatrix} h_i^{(1,1)}(t) & \dots & h_i^{(1,c-b)}(t) \\ \vdots & & \vdots \\ h_i^{(c,1)}(t) & \dots & h_i^{(c,c-b)}(t) \end{bmatrix}. \quad (2)$$

Starting from the  $m_s \cdot (c - b)$ -th column,  $\mathbf{H}^T$  has  $J$  ones in each row and  $K$  ones in each column. The value  $m_s$ , called the syndrome former memory, is determined by the maximal width of the nonzero area in the matrix  $\mathbf{H}^T$ , and the associated constraint length is defined as  $\nu_s = (m_s + 1) \cdot c$ .

In practical applications, periodic syndrome former matrices are of interest. Periodic syndrome formers are said to have a period  $T$  if they satisfy  $\mathbf{H}_i^T(t) = \mathbf{H}_i^T(t + T)$ ,  $i = 0, 1, \dots, m_s$ ,  $t \in \mathbb{Z}$ .

### III. ENCODING OF LDPC CONVOLUTIONAL CODES

In this section, we review the systematic encoder structure of [12] for an LDPC convolutional code of rate  $R = b/c$ . Let

$$\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_t) \quad (3)$$

be the information sequence, where  $\mathbf{u}_t = (u_t^{(1)}, \dots, u_t^{(b)})$ ,  $u_t^{(i)} \in \text{GF}(2)$ ,  $i = 1, \dots, b$ . This sequence is mapped by a convolutional encoder into the code sequence

$$\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_t), \quad (4)$$

where  $\mathbf{v}_t = (v_t^{(1)}, \dots, v_t^{(c)})$ ,  $v_t^{(i)} \in \text{GF}(2)$ ,  $i = 1, \dots, c$ . As a result of the convolutional code structure, a codeword  $\mathbf{v}$  must satisfy

$$\mathbf{v}_t \mathbf{H}_0^T(t) + \mathbf{v}_{t-1} \mathbf{H}_1^T(t) + \dots + \mathbf{v}_{t-m_s} \mathbf{H}_{m_s}^T(t) = \mathbf{0}, \quad t \in \mathbb{Z}. \quad (5)$$

Assuming that all submatrices  $\mathbf{H}_0^T(t)$ ,  $t \in \mathbb{Z}$ , along the diagonal are full rank, this condition can be used to define a shift-register based encoder. The encoder is systematic if the last  $c - b$  rows of the submatrices  $\mathbf{H}_0^T(t)$  are linearly independent, and in this case the first  $b$  symbols of  $\mathbf{v}_t$  coincide with the information symbols, i.e.,  $v_t^{(j)} = u_t^{(j)}$ ,  $j = 1, \dots, b$ .

As a special case of this structure, when the last  $c - b$  linearly independent rows of  $\mathbf{H}_0^T(t)$  are chosen as the  $(c - b) \times (c - b)$  identity matrix, the calculation of the parity symbols is simplified and the encoding equations

are defined using (5) by the expressions

$$v_t^{(j)} = u_t^{(j)}, \quad j = 1, \dots, b, \quad (6)$$

$$v_t^{(j)} = \sum_{k=1}^b v_t^{(k)} h_0^{(k,j-b)}(t) + \sum_{i=1}^{m_s} \sum_{k=1}^c v_{t-i}^{(k)} h_i^{(k,j-b)}(t), \quad j = b + 1, \dots, c. \quad (7)$$

The corresponding shift-register based encoder is shown in Figure 1 for the case  $R = b/c = 1/2$ . Such an encoder realization requires  $c \cdot m_s + b$  memory units and the encoding complexity per encoded bit is proportional to  $(K - 1)$ , independent of the codeword length and the syndrome former memory  $m_s$ . A straightforward encoder for a length  $N$  LDPC block code that multiplies the information sequence by the generator matrix has a complexity per encoded bit that is proportional to the block length  $N$ . Therefore, the special structure of the convolutional code gives it a clear advantage compared to LDPC block codes in terms of encoding complexity.

### IV. IRREGULAR LDPC CONVOLUTIONAL CODE CONSTRUCTION

We concentrate on irregular LDPC convolutional code constructions of rate  $R = b/c = 1/2$  with syndrome former matrix period  $T = m_s$ . However, the ideas presented in this section can readily be applied to any irregular LDPC convolutional code construction of rate  $R = b/c$ ,  $b < c$ , with  $b, c \in \mathbb{Z}$ .

In order to maintain the fast encoding property, the submatrix  $\mathbf{H}_0^T(t)$  must have a  $(c - b) \times (c - b)$  identity matrix in its last  $(c - b)$  rows. For rate  $R = b/c = 1/2$ ,  $\mathbf{H}_0^T(t)$  has size  $c \times (c - b) = 2 \times 1$ . The choice of  $\mathbf{H}_0^T(t) = [1 \ 1]^T$  (as opposed to  $[0 \ 1]^T$ ) simplifies the code construction problem.

Irregular LDPC code ensembles are described in terms of their degree distributions in the Tanner graph representation. For a code with maximum left (right) degree  $d_v$  ( $d_c$ ), the degree distribution polynomials are given by

$$\lambda(X) = \sum_{i=2}^{d_v} \lambda_i \cdot X^{i-1}, \quad (8)$$

$$\rho(X) = \sum_{i=2}^{d_c} \rho_i \cdot X^{i-1}. \quad (9)$$

Here, for an edge (node) perspective degree distribution,  $\lambda_i$  denotes the fraction of edges (nodes) of degree  $i$  among the variable nodes, whereas  $\rho_i$  denotes the fraction of edges (nodes) of degree  $i$  among the check

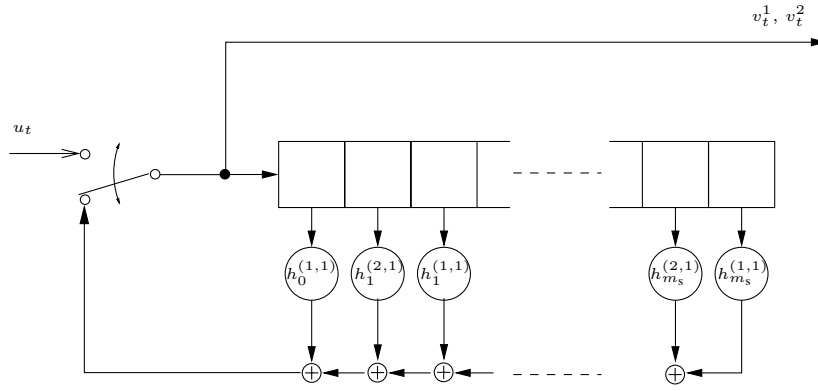


Fig. 1. Shift-register based encoder.

nodes. While the node perspective degree distribution is more intuitive for code construction purposes, edge perspective degree distributions are widely employed in the literature, since they can be used to calculate many important properties of the code ensemble.

In order to construct an LDPC convolutional code with a given syndrome former memory  $m_s$  and a desired edge perspective degree distribution  $(\lambda_C^e, \rho_C^e)$ , we must construct a rate  $R = 1/2$  LDPC block code of length  $m_s \cdot c = 2m_s$  with a node perspective degree distribution  $(\lambda_B^n(X), \rho_B^n(X))$ , as described below. First, the node perspective degree distribution  $(\lambda_C^n, \rho_C^n)$  is obtained from  $(\lambda_C^e, \rho_C^e)$ . Then the left degree distribution polynomial  $\lambda_B^n(X)$  is obtained by dividing  $\lambda_C^n(X)$  by  $X$  in order to decrease all left node degrees by one. In order to maintain a rate of  $R = 1/2$ , the right degree distribution polynomial  $\rho_B^n(X)$  is obtained by dividing  $\rho_C^n(X)$  by  $X^2$ . Thus,  $(\lambda_B^n(X), \rho_B^n(X))$  is calculated as:

$$\lambda_B^n(X) = \lambda_C^n(X)/X, \quad (10)$$

$$\rho_B^n(X) = \rho_C^n(X)/X^2. \quad (11)$$

Once the LDPC block code is constructed (see Figure 2a), its syndrome former matrix is then cut along the diagonal in steps of size  $c \times (c - b) = 2 \times 1$ , and the lower diagonal part is appended to the right hand side of the matrix (see Figure 2b). The resulting matrix is one period of the syndrome former matrix of an LDPC convolutional code. We then append the  $2 \times 1$  subblock  $[1 \ 1]^T$  to the left of every pair of rows (see Figure 2c). The addition of this subblock ensures that each  $\mathbf{H}_0^T(t)$  in the resulting matrix has full rank and that the desired degree distribution is achieved. The resulting matrix is then repeated periodically in order to achieve the infinite size syndrome former matrix of an LDPC convolutional code (see Figure 2d).

A simple example of the construction technique de-

scribed above is given in Figure 2. For ease of representation, an only slightly irregular degree distribution with a very small syndrome former memory ( $m_s = 4$ ) has been chosen. The desired edge perspective degree distribution is given as

$$\lambda_C^e(X) = 0.4286X^2 + 0.5714X^3,$$

$$\rho_C^e(X) = X^6.$$

The corresponding node perspective degree distribution can be calculated as

$$\lambda_C^n(X) = 0.5000X^2 + 0.5000X^3,$$

$$\rho_C^n(X) = X^6.$$

We then construct an LDPC block code of length  $m_s \cdot c = 4 \cdot 2 = 8$ . The node perspective degree distribution for this code is given by

$$\lambda_B^n(X) = \lambda_C^n(X)/X = 0.5000X + 0.5000X^2,$$

$$\rho_B^n(X) = \rho_C^n(X)/X^2 = X^4.$$

The constructed code is shown in Figure 2a. In the next step the lower diagonal portion is cut and appended to the right side, as shown in Figure 2b. The desired  $\mathbf{H}_0^T(t)$  subblock, namely  $[1 \ 1]^T$ , is also appended to each pair of rows of the matrix to maintain the fast encoding property (Figure 2c). The resulting matrix can be used as the syndrome former matrix of an irregular LDPC convolutional code by repeating itself periodically with period  $T = m_s$ . The code construction process is now complete and the resulting syndrome former is shown in Figure 2d.

There are two remarks to be made regarding this construction. The first one pertains to the construction of the LDPC block code. Any known code construction technique for irregular LDPC block codes can be em-

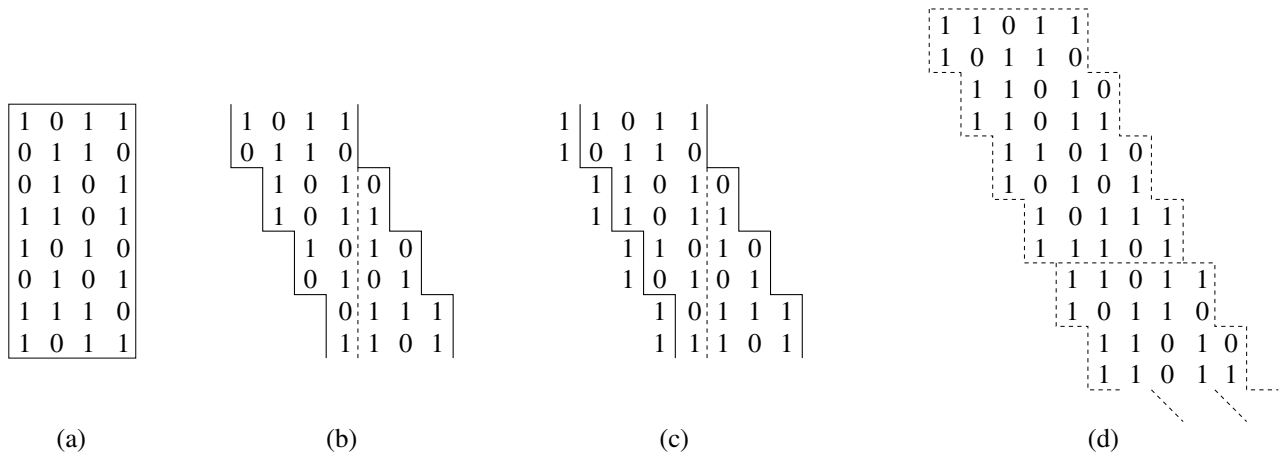


Fig. 2. Irregular LDPC convolutional code construction example for  $m_s = 4$ .

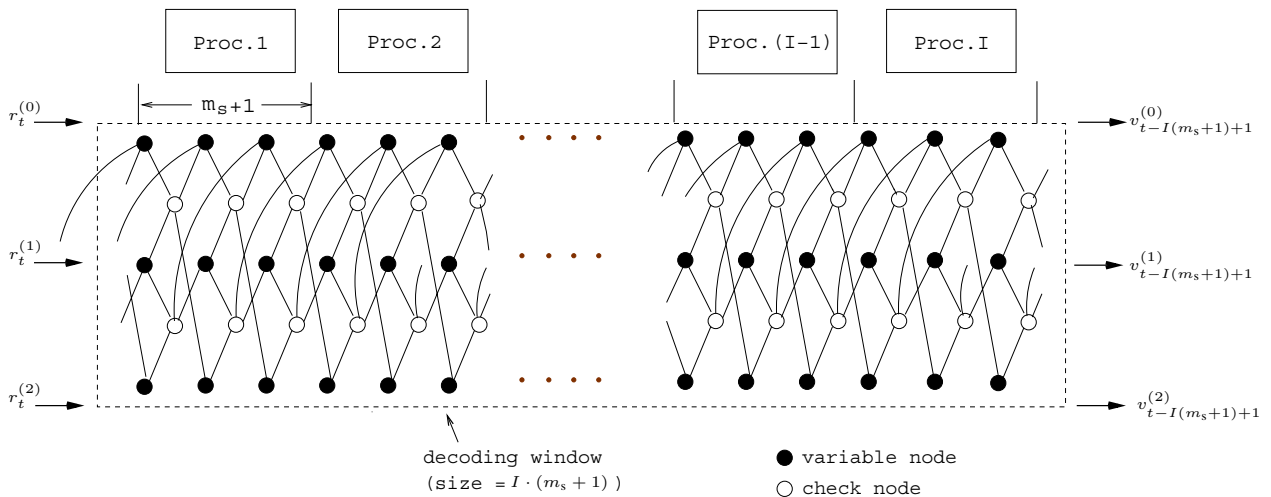


Fig. 3. Tanner graph of  $R=1/3$  LDPC convolutional code and illustration of pipeline decoding.

ployed in this step. However, as will be seen in the next section, the syndrome former memories of interest are quite small compared to the LDPC block code lengths widely employed in the literature. Small syndrome former memory  $m_s$  corresponds to a small Tanner graph, and it is a well-known problem with constructing short irregular block codes that the Tanner graph has small girth. In order to avoid such a situation, we employ the progressive edge growth (PEG) [16] algorithm to maximize girth.

The other point to be made is in regard to the minimum degree of a variable node in the Tanner graph of the LDPC block code. If the desired irregular degree distribution has degree 2 variable nodes, as a result of (10),  $\lambda_B^n(X)$  has a term that corresponds to degree 1 variable nodes in the Tanner graph. Although there are no LDPC block codes with this property in the literature, most of the known code construction techniques for

LDPC block codes – including random constructions and the PEG algorithm – manage to construct such codes without modification.

## V. SIMULATION RESULTS

The performance of irregular LDPC convolutional codes with moderate constraint lengths is considered in this section. A memory of  $m_s = 1199$  was chosen for the simulations. For the irregular convolutional codes, the optimum degree distribution for rate  $R = 1/2$  irregular LDPC block codes over an additive white Gaussian noise (AWGN) channel was chosen [10]:

$$\begin{aligned} \lambda_C^e(X) &= 0.21991X + 0.23328X^2 + 0.02058X^3 \\ &\quad + 0.08543X^5 + 0.06540X^6 + 0.04767X^7 \\ &\quad + 0.01912X^8 + 0.08064X^{18} + 0.22798X^{19}, \\ \rho_C^e(X) &= 0.64854X^7 + 0.34747X^8 + 0.00399X^9. \end{aligned}$$

The LDPC convolutional codes were decoded by a sliding window pipeline decoder [12] that employs an on-demand variable node updating schedule [17].

Although the Tanner graph corresponding to an LDPC convolutional code has an infinite number of nodes, the distance between two variable nodes that are connected to the same check node is limited by the memory of the code. This allows continuous decoding with a decoder that operates on a finite window sliding along the received sequence, similar to a Viterbi decoder with finite path memory [18]. The decoding of two variable nodes that are at least  $(m_s + 1)$  time units apart can be performed independently, since the corresponding bits cannot participate in the same parity-check equation. This allows the parallelization of the  $I$  iterations by employing  $I$  independent identical processors working on different regions of the Tanner graph simultaneously. A pipeline decoder that is based on this idea was introduced by Jiménez Felström and Zigangirov in [12]. The operation of this decoder on the Tanner graph for a simple time-invariant rate  $R = 1/3$  LDPC convolutional code with  $m_s = 2$  is illustrated in Figure 3.

When the first  $c$  received symbols of the transmitted sequence enter the pipeline decoder, it takes  $I \cdot (m_s + 1)$  time units for this  $c$ -tuple to reach the output of the window so that a decision on these symbols can be made. Once this initial decoding delay has elapsed, the decoder produces a continuous output stream, i.e., at each time unit,  $c$  newly received values enter the decoder and  $c$  decoded bits leave it. The  $I$  processors perform the  $I$  decoding iterations simultaneously. At each time unit, the  $i$ -th processor begins by activating the first column of  $(c-b)$  check nodes in its operating region and then proceeds to activate the last column of  $c$  variable nodes that are about to leave the operating region. A check node activation is the step where the check node collects all the incoming messages from its neighboring variable nodes, calculates the outgoing messages, and sends the new messages to each neighboring variable node. Correspondingly, during a variable node activation, all the incoming messages to a variable node from the neighboring check nodes are collected and the new outgoing messages are calculated. Then the new messages are sent to each neighboring check node.

A maximum number of iterations of 100 was allowed for each of the simulations, and decoder stopping rules (see [17]) were employed. For BER performance comparison purposes, corresponding LDPC block codes were chosen with block length  $N = 2400$  in order to have the same processor (hardware) complexity for

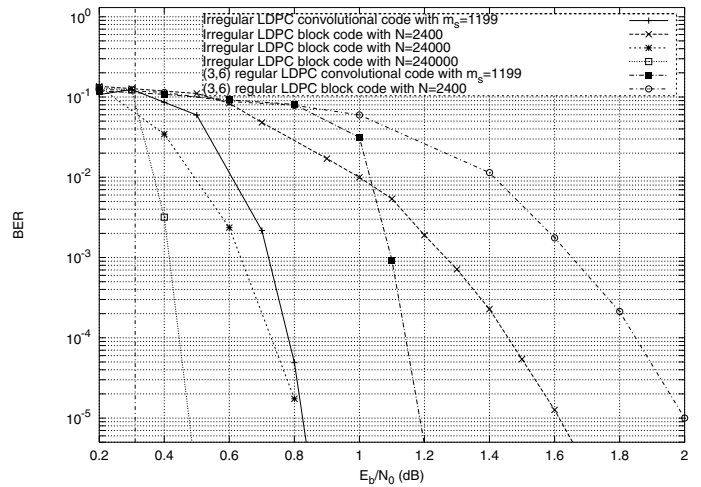


Fig. 4. BER performance of the proposed irregular LDPC convolutional codes.

decoding [19]. Longer irregular LDPC block codes with block lengths  $N = 24000$  and  $N = 240000$ , implying greater processor complexity by a factor of 10 or 100, respectively, were also simulated.

Performance curves for an AWGN channel are presented in Figure 4. At a BER of  $10^{-5}$ , the irregular LDPC convolutional code with syndrome former memory  $m_s = 1199$  outperforms the corresponding irregular LDPC block code of length  $N = 2400$  with the same processor complexity by almost 0.80 dB. Longer irregular LDPC block codes of lengths  $N = 24000$  and  $N = 240000$  achieve performance closer to that of the irregular LDPC convolutional code, but with much greater processor complexity. The irregular LDPC block code of length  $N = 24000$  performs approximately the same as the irregular LDPC convolutional code, whereas the irregular LDPC block code of length  $N = 240000$  outperforms both of them by 0.30 dB. The iterative decoding threshold for the irregular degree distribution is also marked in Figure 4 at 0.3104 dB. Even though the LDPC convolutional code decoder requires  $I$  processors, compared to only one for the LDPC block code decoder, from a circuit design perspective, it is much simpler to design a small processor and then implement  $I$  copies than to design and implement a much larger processor. Further discussion on the topic can be found in [19]. Finally, we see that the irregular LDPC convolutional code also outperforms corresponding regular LDPC block and convolutional codes.

## VI. CONCLUSIONS

LDPC convolutional codes retain all of the desirable features of LDPC block codes, and a sliding window

message passing decoder can be used to decode LDPC convolutional codes with large constraint lengths. It has been observed [12], [20] that the BER performance of rate  $b/c$  LDPC convolutional codes with constraint length  $\nu_s = (m_s + 1) \cdot c$  is superior to that of rate  $k/n = b/c$  LDPC block codes with block length  $n = \nu_s$ . Also, Gallager in [2] showed that there exist LDPC block codes with distance increasing linearly with block length. An analogous result on free distance for LDPC convolutional codes, i.e., the free distance increases linearly with constraint length, has recently been proved in [21], [22]. Further, the asymptotic slope of free distance vs constraint length for  $(J, K)$  regular LDPC convolutional codes is larger than the corresponding parameter for  $(J, K)$  LDPC block codes.

Given their excellent BER performance, LDPC convolutional codes are ideal candidates for practical low-complexity communication scenarios. In this paper, we proposed a novel code construction algorithm for designing irregular LDPC convolutional codes. The constructed codes have a fast encoding property that allows them to be encoded using a shift-register based encoder with very low complexity. For moderate constraint length codes, computer simulations show that irregular LDPC convolutional codes perform very well compared to irregular LDPC block codes with comparable processor complexity.

#### ACKNOWLEDGEMENT

This work was supported in part by NSF Grant CCR02-05310 and NASA Grant NNG05GH73G.

#### REFERENCES

- [1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: turbo codes," in *Proc. of the IEEE International Conference on Communications*, (Geneva, Switzerland), pp. 1064–1070, May 1993.
- [2] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
- [3] N. Wiberg, *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Sweden, 1996.
- [4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, August 1996.
- [5] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proc. 30th Annual ACM Symp. Theory of Computation*, pp. 249–258, May 1998.
- [6] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 585–598, Feb. 2001.
- [7] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 599–618, Feb. 2001.

- [8] S. Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 657–670, Feb. 2001.
- [9] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
- [10] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 619–637, Feb. 2001.
- [11] S. Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Communications Letters*, vol. 5, pp. 58–60, Feb. 2001.
- [12] A. Jiménez-Feltström and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 2181–2191, Sept. 1999.
- [13] S. Bates, Z. Chen, and X. Dong, "Low-density parity check convolutional codes for Ethernet networks," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, (Victoria, B.C., Canada), Aug. 2005.
- [14] S. Bates, D. Elliot, and R. Swamy, "Termination sequence generation circuits for low-density parity-check convolutional codes," *submitted to IEEE Trans. Circuits and Systems I*, March 2005.
- [15] S. Bates, L. Guntorphe, A. E. Pusane, Z. Chen, K. Sh. Zigangirov, and D. J. Costello, Jr., "Decoders for low-density parity-check convolutional codes with large memory," in *Proc. 12th NASA Symposium on VLSI Design*, (Coeur d'Alene, Idaho, U.S.A.), Oct. 2005.
- [16] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Global Telecommun. Conf.*, (San Antonio, TX), pp. 995–1001, Nov. 2001.
- [17] A. E. Pusane, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Reduced complexity decoding strategies for LDPC convolutional codes," in *Proc. IEEE Intl. Symposium on Information Theory*, (Chicago, Illinois, USA), p. 490, June 2004.
- [18] S. Lin and D. J. Costello, Jr., *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 2nd ed., 2004.
- [19] A. E. Pusane, A. Jiménez-Feltström, A. Sridharan, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Implementation aspects of LDPC convolutional codes," *submitted to IEEE Trans. Commun.*, Nov. 2005.
- [20] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inform. Theory*, vol. IT-50, pp. 2966–2984, Dec. 2004.
- [21] A. Sridharan, D. Truhachev, M. Lentmaier, D. J. Costello, Jr., and K. Sh. Zigangirov, "On the free distance of LDPC convolutional codes," in *Proc. IEEE Intl. Symposium on Information Theory*, (Chicago, Illinois, USA), p. 311, June 2004.
- [22] A. Sridharan, D. Truhachev, M. Lentmaier, D. J. Costello, Jr., and K. Sh. Zigangirov, "Distance bounds for a class of LDPC convolutional codes constructed from permutation matrices," *submitted to IEEE Trans. Inform. Theory*, Jan. 2005.