Turbo Product Codes Based on Convolutional Codes

Orhan Gazi and Ali Özgür Yılmaz

In this article, we introduce a new class of product codes based on convolutional codes, called convolutional product codes. The structure of product codes enables parallel decoding, which can significantly increase decoder speed in practice. The use of convolutional codes in a product code setting makes it possible to use the vast knowledge base for convolutional codes as well as their flexibility in fast parallel decoders. Just as in turbo codes, interleaving turns out to be critical for the performance of convolutional product codes. The practical decoding advantages over serially-concatenated convolutional codes are emphasized.

Keywords: Iterative decoding, convolutional product codes, interleaving, puncturing.

I. Introduction

One of the most successful works to approach the Shannon limit was published in 1993 by C. Berrou, A. Glavieux, and P. Thitimajshima [1]. They introduced turbo codes, also known as parallel concatenated convolutional codes (PCCC). Turbo codes can achieve bit error rate (BER) levels of around 10^{-5} at code rates quite close to the corresponding capacity with reasonable decoding complexity. The use of soft-in soft-out decoding algorithms was a key to this success. In the last decade, similar codes such as serially-concatenated convolutional codes (SCCCs) [2], low-density parity check codes [3], and block product codes have been extensively studied. The studies on product codes were initiated by Elias [4]. Product codes enjoy a high degree of parallelization as opposed to many other forms of concatenated code structures, for example, PCCC.

The product codes studied thus far have been constructed using linear block codes, such as Hamming, extended Hamming [5], [6], BCH [7], [8], and Reed Solomon [9] codes. Single parity check (SPC) product codes are studied in [10]. Three and more dimensional SPC product codes are studied in [11]. Product codes have also attracted practical attention lately. Digital signal processing and field-programmable gate array implementations are studied in [12]. Product codes are traditionally constructed by linear block codes. Block codes have a trellis structure with a time varying property [13]. The product code we propose in this paper is constructed by using time-invariant convolutional codes. Its component codes' trellis structure does not vary in time as in product codes constructed with Hamming, extended Hamming, BCH, and Reed Solomon block codes. Moreover, the number of states in the trellis structure of a block code may grow exponentially with the difference of codeword and data block lengths [13], whereas the number of states in a

Manuscript received Sept. 25, 2005; revised Apr. 26, 2006.

This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 104E027.

Orhan Gazi (phone: + 90 312 284 4500/4015, email: o.gazi@ari.cankaya.edu.tr) is with Department of Electronic and Communication Engineering, Cankaya University, Ankara, Turkey.

Ali Özgür Yılmaz (email: aoyilmaz@eee.metu.edu.tr) is with Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, Turkey.

convolutional code can be set as desired. The time invariant trellis structure of convolutional codes makes them more convenient for implementation. In addition, numerous practical techniques such as trellis coded modulation and puncturing can be simply utilized with convolutional codes as opposed to linear block codes. A code from the same family was previously studied for orthogonal frequency-division multiplexing in [14] but was not analyzed or further elaborated.

Multi-input multi-output (MIMO) techniques are quite important to enhance the capacity of wireless communication systems. Space-time trellis codes provide both diversity and coding gain in MIMO channels and are widely used [15]. Space-time trellis codes usually have time-invariant trellis structures just like convolutional codes. Thus, a product code based on convolutional codes is more suitable for integration with MIMO channels and poses an alternative to block product codes.

Due to these advantages of convolutional codes, we propose a class of product codes constructed by using convolutional codes, which we call convolutional product codes (CPCs). In this paper, we will investigate the factors that affect the performance of CPCs and leave the issues regarding space time trellis codes to other publications.

The outline of the paper is as follows. The proposed code structure and the decoding algorithm for CPCs are given in section II. The minimum distance of these codes is studied in section III. In section IV, implementation advantages of CPCs are given. Simulation results are presented in section V. Concluding remarks are given in section VI.

II. CPC Encoder and Decoder

1. CPC Encoder

A regular product code is constructed by placing the information bits/symbols into a matrix. The rows and columns are encoded separately using linear block codes [5]-[8]. This type of a product encoder is shown in Fig. 1. It is seen from the figure that the data and parity bits are grouped separately.

In our case, we use convolutional codes instead of linear block codes to encode rows and columns. This is illustrated in Fig. 2. When compared to Fig. 1, it is obvious that data and parity bits are mixed uniformly.

Encoding is performed by using a matrix that determines how each encoder works. The data to be sent is put into the matrix. Each row of the matrix is encoded using a convolutional code. We use the same recursive systematic convolutional code to encode each row, although different convolutional codes can be used for this purpose. Once each row is encoded, the matrix is sent, if desired, to an interleaver. Our data matrix dimension is $k \times k$, and the encoded data matrix dimension is $n \times n$, that is, our code is an $(n \times n, k \times k)$ code. The interleaved matrix is coded column-wise. In our simulation we used the rate 1/2 recursive systematic convolutional code with the matrix generator $(1, 5/7)_{octal}$ to encode each row and column. Hence, the overall code rate is 1/4. The general encoding procedure, which includes any type of interleaver, is illustrated in Fig. 3.

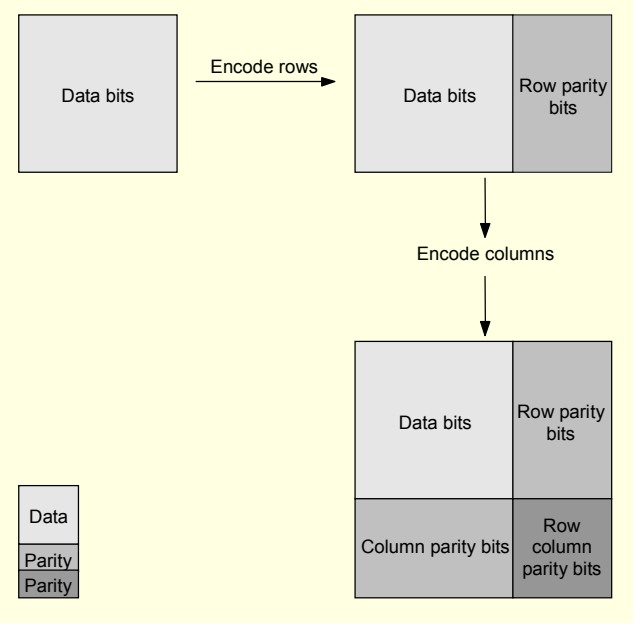


Fig. 1. Regular product code encoding procedure, where a block code is used to encode rows and columns.

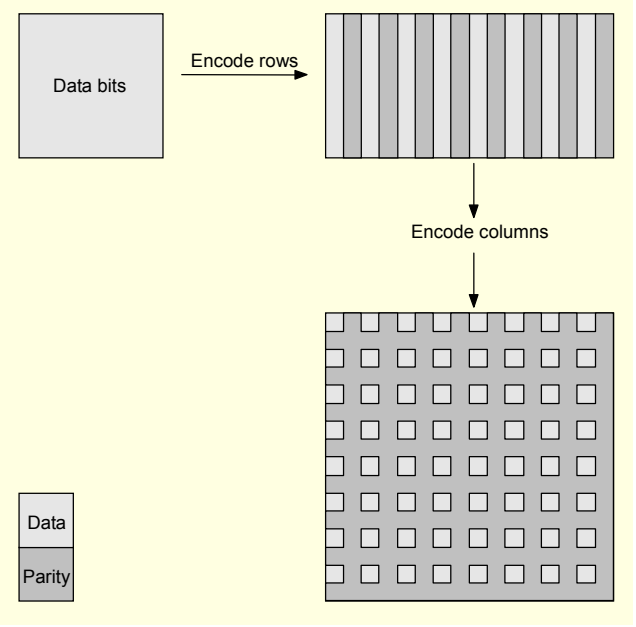


Fig. 2. CPC encoding procedure without an interleaver.

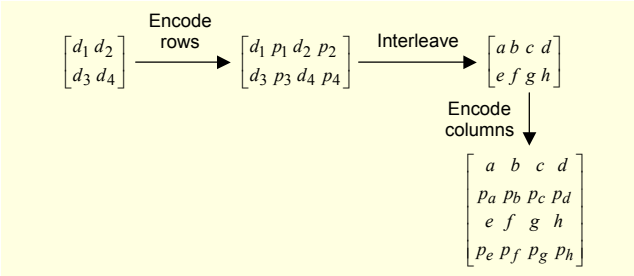


Fig. 3. Convolutional product code encoder with any type of interleaver (d denotes data bits and p denotes parity bits).

2. CPC Decoder

Convolutional product coded data is multiplexed to a single stream and binary phase shift key (BPSK) modulated. The BPSK-modulated signal is passed through an additive white Gaussian noise channel with double-sided noise power spectral density $N_0/2$, that is, the noise variance is $\sigma^2 = N_0/2$. We used the log-MAP soft decoding algorithm [16], [17] to iteratively decode the convolutional product code. Since columns were encoded last, each column is independently decoded one by one. The extrinsic information obtained from the columns is passed to the row decoder after being de-interleaved. Then, row decoding proceeds; rows are decoded one by one, and interleaved extrinsic information is passed to the column decoder. The CPC decoding procedure is depicted in Fig. 4. This procedure is repeated for a sufficient number of times. The decoding structure employed in this study is the same as that of serially-concatenated codes in Fig. 5 [17]. For frames of equal length, an SCCC decoder uses two log-MAP decoders and performs quite well at low rates. CPC decoders can utilize

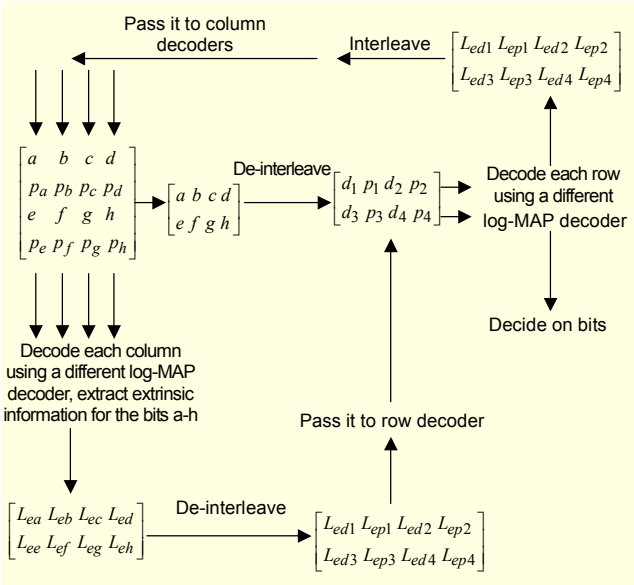


Fig. 4. Decoding operation of the convolutional product code.

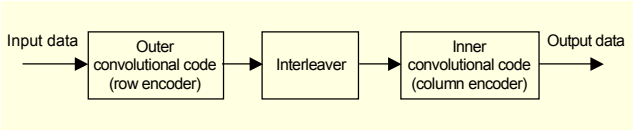


Fig. 5. SCCC encoding operation.

many log-MAP decoders in-parallel, thus showing smaller decoding delays. Therefore, we will compare the proposed CPC structure to that of SCCC.

3. Puncturing

Puncturing is a widely used tool to increase the code rate of convolutional codes [18]. The puncturing operation increases the rate of a code, but decreases the free distance. This results in a worse error rate performance compared to a non-punctured case. We used the puncturing matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

to puncture the convolutional component codes.

We studied two cases. In the first case, puncturing is applied only to the column encoders, resulting in a code rate of 2/3 each. The overall code rate becomes $(1/2)\times(2/3)=1/3$. When trellis termination is used for rows and columns, a convolutional code with a slightly smaller overall code rate ($\leq 1/3$) is produced. In the other case, we apply puncturing to each row and column encoder, resulting in an increased code rate of approximately $(2/3)\times(2/3)=4/9$. Simulation results for punctured convolutional product codes (PCPCs) will be presented in section V.

III. CPC Minimum Distance and its Asymptotic Performance

The Hamming weight of a binary codeword is defined as the number of '1's available in the codeword [13]. The minimum distance of a linear code is the minimum Hamming weight of all the codewords. The minimum distance plays an important role in the code performance. As it gets larger, code performance improves, especially at high signal-to-noise ratio (SNR) values [13]. We assume that d_{free} is the free distance of the component convolutional codes used in CPCs with trellis termination. We will investigate the minimum distance of the CPCs according to the usage of the interleavers.

1. No Interleaving

After the first stage of the CPC encoding operation (row encoding), it is obvious that one of the rows of the row-

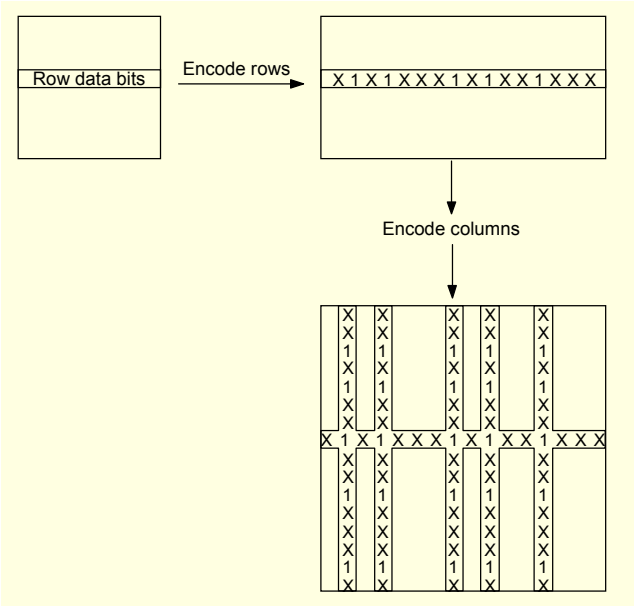


Fig. 6. If the column elements are not mixed, d^2_{free} is preserved.

encoded matrix should contain at least d_{free} number of '1's. This means that there are d_{free} columns containing at least a single '1' in the row-encoded matrix. When columns are encoded, there exists at least d_{free} number of columns each containing at least d_{free} '1's. Hence, in total there are at least d^2_{free} '1's in the coded matrix [6]. This is the d_{min} distance of the CPC whose component convolutional codes have a trellis termination constraint. In Figs. 6 and 7, this concept is explained for $(1, 5/7)_{octal}$ component convolutional codes whose free distance is 5. In summary, if no interleaver is used, the CPC minimum distance is d^2_{free}.

2. Column S-Random Interleaver

Both to preserve the d^2_{free} minimum distance of the CPC, and to benefit from the interleaving gain, after row encoding we used S-random interleavers for each column, that is, each column is interleaved but different column elements are not mixed. In this way, we guarantee that d_{free} number of columns contains a single '1' before column encoding operation. We call this type of interleaving column S-random interleaving to distinguish it from regular S-random interleaving. A helical interleaver [19] also does not mix the different column elements. A helical interleaver and a combination of helical and column S-random interleavers will also be considered.

3. Full S-Random Interleaver

If an S-random interleaver is used for all the elements of a matrix after row encoding, the number of columns that contain

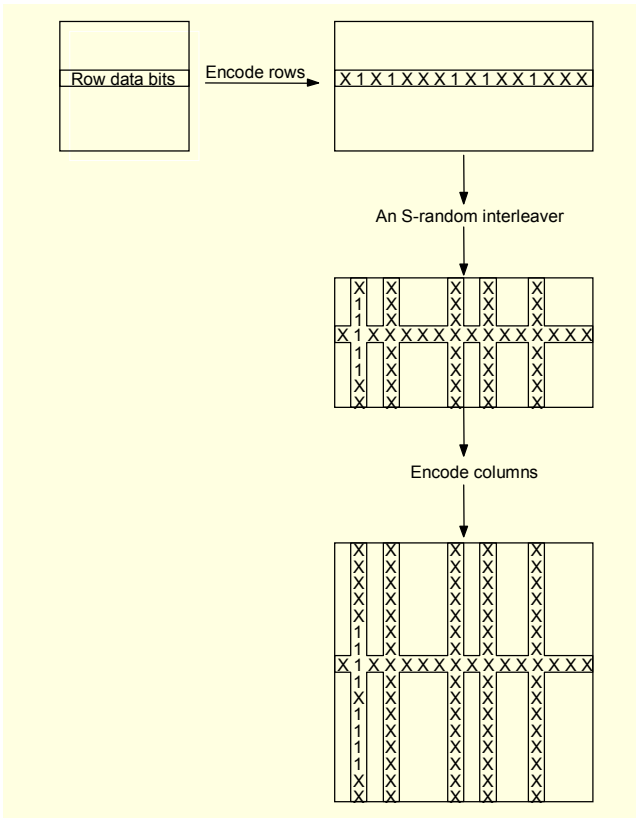


Fig. 7. d^2_{free} is not preserved if an S-random interleaver is used ('x' stands for a single or a group of 0's).

a single '1' is not necessarily equal to d_{free}=5. As a result, the CPC minimum distance is no longer necessarily equal to d^2_{free}. In fact, after an interleaving operation, all the '1's may appear in a single column. This means that the CPC minimum distance is lower bounded by d_{free}. We call this type of interleaving full S-random interleaving. In Fig. 7, the effect of the full S-random interleaver is illustrated. It is seen from Fig. 7 that when the row-encoded matrix is S-random interleaved, all the '1's appearing in a row may go to a single column. This verifies that the CPC minimum distance is lower bounded by d_{free}.

4. Punctured CPCs

The puncturing operation decreases the free distance of convolutional codes. In our case, we puncture the $(1, 5/7)_{octal}$ component convolutional code that has d_{free}=5, that is, an input sequence '0111' produces minimum Hamming weight codeword '00111011'. When the puncturing matrix is applied, its free distance decreases to d_{free}=3, that is, deleting every second parity bit in a periodic manner, '001x101x' is obtained from the minimum Hamming weight codeword. Hence, the CPCs constructed using punctured component convolutional codes have a smaller minimum distance. In fact,

the minimum distance is equal to $d_{\text{free}}^2 = 9$ if no interleaving operation is performed or a column S-random interleaver is used.

5. Asymptotic Performance

If row and column convolutional codes are trellis terminated, the row and column convolutional codes can be considered as block codes. Asymptotic performance studies made for block product codes are also valid for a convolutional product code. The BER probability of the CPCs can be approximated using the formula,

$$P_b \approx \frac{N_{c,d_{free}}^2 \, w_{c,d_{free}}^2}{k^2} Q\left(\sqrt{d_{free}^2 \frac{2E_s}{N_0}}\right), \; E_s = r^2 E_b, \qquad (1)$$

where d_{free} is the free distance of the convolutional code used to construct the convolutional product code, $N_{c,d_{\text{free}}}$ is the number of convolutional codewords with free distance d_{free}, k is the length of the data information frame, and $w_{c,d_{\text{free}}}$ is the average Hamming weight of the information words that produces convolutional codewords with Hamming weight d_{free}. Also, r is the rate of the component convolutional codes and is equal to 1/2 or 2/3 in our case, and Q is the error function given as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{\pi}^{\infty} e^{\frac{-t^2}{2}} dt.$$

The BER approximation in (1) is valid if no interleaver is used during the CPC encoding operation. If an interleaver is used, it does not hold anymore.

IV. Practical Implementation Advantages

The implementation advantage of CPC will be discussed herein with the parameters used in this study. Trellis termination will be neglected in calculation and will not alter the results significantly. In SCCC, for a given transmit data vector of length L, two log-MAP decoders are needed. The first decoder has a complexity of order O(2L)[1)] and a time delay of O(2L). The second decoder has a shorter input, thus it has a complexity of O(L) and a time delay of O(L). In total, the complexity is of O(3L) and the time delay is of O(3L). In CPC, columns are decoded first. The use of separate log-MAP decoders for each row and column makes parallel processing operations possible. Each column decoder has a complexity of O($\sqrt{L}$) and time delay of O($\sqrt{L}$). Since these decoders are run in-parallel, the total column decoding complexity is of O(2L) but the time delay is of O($\sqrt{L}$). Similarly, row decoding has a total complexity of O(L) and time delay of O($\sqrt{L}$). Hence, although both complexities are the same, time delays differ very much and bring about an O($\sqrt{L}$)-time increase in decoding rate. Hence, the main advantage of CPCs lies in their suitability for a parallel decoding procedure. Although there are some proposed methods for the parallel decoding of SCCCs and PCCCs, these methods usually propose extra algorithms to solve problems met in parallel processing. Such algorithms not only bring extra complexity to the decoding operation [20]-[21], but also may suffer from performance loss. This situation is totally remedied with the proposed CPCs.

1) The meaning of O(L) here is different from its conventional usage. By O(L) we mean that the computation load is proportional to L, i.e., computation amount is approx. kxL where k is the number of parameters to be computed in a single stage of the code trellis.

V. Simulation Results

1. Interleaving Effects

A. No Interleaver

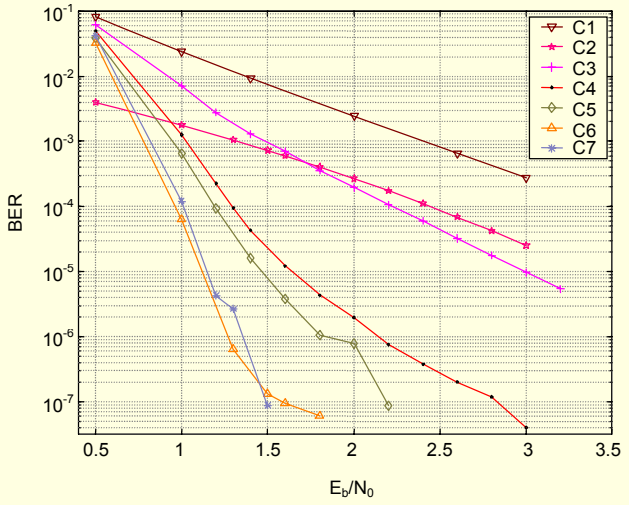
In this case, no interleaving operation is performed after row encoding. Trellis termination bits are added both to rows and columns. The minimum distance of the CPC is $d_{\min} = d_{\text{free}}^2 = 25$. Trellis termination bits are necessary to guarantee $d_{\min} = d_{\text{free}}^2$; otherwise $d_{\min}$ is not equal to d_{free}^2 anymore. The performance graph of this code is shown in Fig. 8. It can be seen from the graph that the performance of this CPC is not good for low SNR values, although its minimum distance is large. As is well known, the minimum distance dominates the performance of the code at high SNR values.

B. Full S-Random Interleaver

After a row-encoding operation, an S-random interleaver (S = 18) is used. We also simulated a serially-concatenated convolutional code to compare against CPC due to the similarity of the code structure and good performance at low rates. The performance graph is seen in Fig. 8. As seen from the performance curve, the performance is very good compared to the cases where interleavers other than S-random are used. Due to the S-random interleaver used after row encoding, the minimum distance of the CPC is not necessarily equal to d_{free}^2. CPC with a full S-random interleaver shows the best performance at low rates due to the large interleaver gain.

C. Column S-Random Interleaver

To obtain both a better performance than that of the no interleaver case and to preserve that $d_{\min} = d_{\text{free}}^2$ of CPC, we applied an S-random interleaver (S = 3) to each column separately. We call such interleaving column S-random



C1: No interleaver is used (rate ≈ 1/4)
C2: Theoretical bound (rate ≈ 1/4)
C3: Helical interleaver is used (rate ≈ 1/4)
C4: Each column is S-random interleaved (column S-random) (rate ≈ 1/4)
C5: Helical + column S-random interleaver is used (rate ≈ 1/4)
C6: Full S-random interleaver is used (rate ≈ 1/4)
C7: SCCC with S-random interleaver (rate ≈ 1/4)

Fig. 8. SCC and CPC performance graph for different interleavers (iteration number = 12, frame length = 1024).

interleaving. Different column elements are not mixed. From Fig. 8, it can be seen that the performance is better compared to the CPC in which no interleaver is used. Its performance is worse than the CPC with which a full S-random interleaver is used after row encoding. The usage of the helical interleaver also guarantees that the minimum distance of CPC equals d^2_{free}. We also investigated the case in which a helical interleaver is followed by a column S-random interleaver. It is seen that such an interleaver results in a slightly better performance than the one where only a column S-random interleaver is used during the encoding procedure.

2. Trellis Termination Effects

We simulated three trellis termination cases where trellis termination bits are added to the rows only (CPC RT), to both rows and columns (CPC TT), and neither to rows nor to columns (CPC No TT). Although the addition of trellis termination bits decreases the code rate, they are critical for good performance of the convolutional product code as seen in Fig. 9. The addition of trellis termination bits in a turbo or serially-concatenated code shows negligible improvement of the code performance [22]. Without trellis termination, the performance of the CPC degrades drastically. The performance graphs are seen in Fig. 9. When only rows are trellis

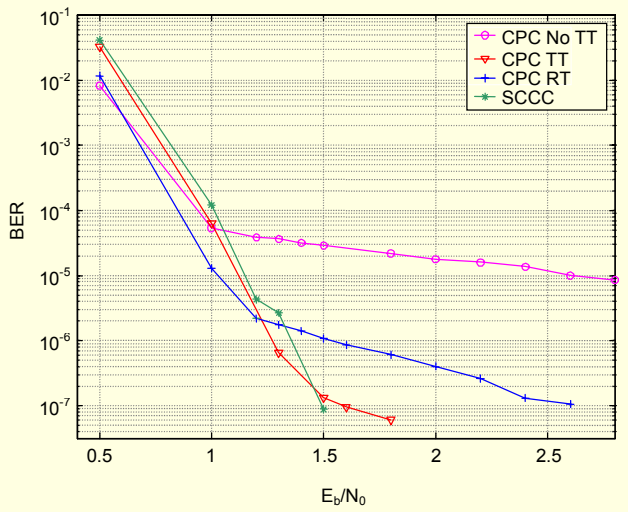
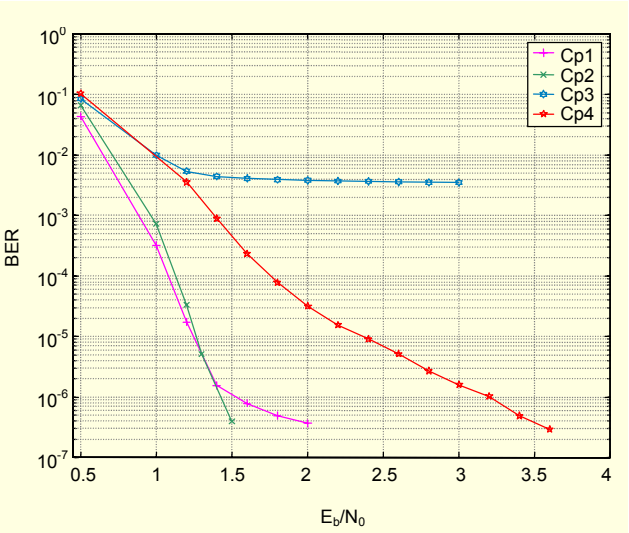


Fig. 9. CPC and SCCC performance graph (frame length = 1024, iteration number = 12).



Cp1: SCCC with S-random interleaver (rate ≈ 1/3)
Cp2: CPC with Full S-random interleaver (rate ≈ 1/3)
Cp3: CPC with Full S-random interleaver (rate ≈ 4/9)
Cp4: Each column is S-random interleaved (column S-random) (rate ≈ 4/9)

Fig. 10. Punctured CPC and punctured SCCC performance graph. CPC rows and columns are trellis terminated. (frame length = 1024, iteration number = 12).

terminated, a convolutional product code has better performance at very low E_b/N_0 levels. However, the BER slope decreases at higher E_b/N_0 levels when compared to the case where both rows and columns are trellis terminated. We see that CPC RT is better than the SCCC and CPC TT at low E_b/N_0 regions. Although it is quite close to BER 10^{-7}, SCCC seems to have an error curve of higher slope compared to CPC TT at higher E_b/N_0 values.

3. Puncturing Effects

Puncturing is first applied only to rows, resulting in a rate 2/3 CPC. From Fig. 10, it is seen that the performance of the CPC with a full S-random interleaver is good after being punctured. When the puncturing process is applied to both rows and columns, it results in a CPC rate of approximately 4/9. From Fig. 10, it is seen that the performance becomes very poor for CPC with a full S-random interleaver. Recall that $d_{\min}$ is not necessarily lower bounded by d_{free}^2 when an S-random interleaver is used. Thus, the particular interleaver we used resulted in a low $d_{\min}$. When $d_{\min} \geq d_{\text{free}}^2$ is ensured by column S-random interleaving, performance is enhanced significantly.

VI. Conclusion

In this article, we studied a new class of product codes based on convolutional codes. This type of product code has component codes with a time invariant trellis structure, as opposed to product codes constructed with linear block codes (Hamming, BCH, Reed Solomon, and so on). Hence, CPC may be more favorable for implementation than linear block product codes. When compared to serially-concatenated convolutional codes, it exhibits comparable BER levels which are of practical interest.

We investigated the effects of different interleavers on the performance of CPCs. It was seen that CPCs are outperformed by other codes unless good interleavers are used. We proposed interleaving methods to preserve the greatest minimum distance of CPCs. It is seen that the performance of a CPC is best at low rates when a full S-random interleaver is used. Column S-random interleavers are much better for punctured CPCs.

Currently, we are investigating the effects of various interleavers and the incorporation of trellis coded modulation in row and column encoding. Since CPCs employ matrices in encoding, it can be easily extended to multi-carrier modulation where the vertical dimension can correspond to the sub-carriers. The approach presented here can be successfully extended to space-time trellis coding. Thus, our future studies will also include a joint structure for CPCs and MIMO space-time frequency codes.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments and helpful corrections.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *Proc. ICC'93*, Geneva, Switzerland, May 1993, pp. 1064-1070.

[2] S. Benedetto, L. Gaggero, R. Garello, and G. Montorsi, "On the Design of Binary Serially Concatenated Convolutional Codes," *Proc. VIII Communication Theory Mini-Conf.* (*CTMC*), Vancouver, BC, Canada, June 1999, pp. 32-36.

[3] R. G. Gallager, "Low Density Parity Check Codes," *IRE Trans. Inform. Theory,* vol. IT-8, Jan. 1962, pp. 21-28.

[4] P. Elias, "Error Free Decoding," *IRE Trans. Inform. Theory,* vol. IT-4, Sept. 1954, pp. 29-37.

[5] E. Hewitt, "Turbo Product Codes for LMDS," *IEEE Radio and Wireless Conf.,* Aug. 1998.

[6] Nam Yul Yu, Young Kim, and Pil Joong Lee, "Iterative Decoding of Product Codes Composed of Extended Hamming Codes," *5th IEEE Symposium on Computers and Communications (ISCC 2000)*, Antibes, France, 04 – 06 July 2000, pp. 732–737.

[7] R. M. Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Trans. Commun.,* vol. 46, no. 8, Aug. 1998, pp. 1003-1010

[8] T. Shohon, Y. Soutome, and H. Ogiwara, "Simple Computation Method of Soft Value for Iterative Decoding of Product Code Composed of Linear Block Code," *IEIC Trans. Fundamentals,* vol. E82-A, no. 10, Oct. 1999, pp. 2199–2203.

[9] Omar Aitsab and Ramesh Pyndiah, "Performance of Reed Solomon Block Turbo Codes," *Proc. IEEE LOBECOM' 96 Conf.,* London, U.K., vol. 1/3, Nov. 1996, pp. 121-125.

[10] David Rankin and T. Aaron Gulliver, "Single Parity Check Product Codes," *IEEE Trans. Commun.,* vol. 49, no. 8, Aug. 2001, pp. 1354-1362.

[11] D. M. Rankin and T. A. Gulliver, "Randomly Interleaved Single Parity Check Product Codes," *Proc. IEEE Int Symp. on Inform. Theory,* June 2000, pp. 88.

[12] A. Goalic and R. Pyndiah, "Real Time Turbo Decoding of Product Codes on a Digital Signal Processor," *Int. Symposium on Turbo Codes and Related Topics,* Brest, Sept. 1997, pp. 624-628.

[13] S. Lin, D. Costello, *Error Control Coding*, Prentice Hall, 2004.

[14] F. Sanzi and S. ten Brink, "Iterative Channel Estimation and Decoding with Product Codes in Multicarrier Systems," *IEEE VTS Fall VTC2000 52nd Vehicular Technology Conf.,* Boston, MA, USA, Sep. 2000, pp. 1388–1344.

[15] V. Tarokh V, N. Seshadri, and A. R. Calderbank, "Space-Time Codes for High Data Rate Wireless Communication: Performance Criterion and Code Construction," *IEEE Trans. Inform. Theory,* vol. 44, no. 2, Mar. 1998, pp. 744-765.

[16] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inform. Theory,* vol. 42, no. 2, Mar. 1996, pp. 429–445.

[17] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serially Concatenation of Interleaved Codes: Design and Performance Analysis," *IEEE Trans. Inform. Theory,* vol. 44, May 1998, pp. 909-926.

[18] J. Hagenauer, "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications," *IEEE Trans. Inform. Theory,* vol. 44, no. 3, May 1998, pp. 909- 926.

[19] Branka Vucetic and Jinhong Yuan, *Turbo Codes: Principles and Applications*, Kluwer Academic Publishers, May 2000.

[20] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping Interleaving Laws to Parallel Turbo and LDPC Decoder Architectures," *IEEE Transactions on Information Theory,* vol. 50, no. 9, Sep. 2004, pp. 2002–2009.

[21] Seokhyun Yoon and Yeheskel Bar-Ness, "A Parallel MAP Algorithm for Low Latency Turbo Decoding," *IEEE Communication Letters*, vol. 6, no. 7, July 2002, pp. 288–290.

[22] P. Robertson, "Illuminating the Structure of Parallel Concatenated Recursive (TURBO) Codes," *Proc. GLOBECOM'94,* San Francisco, CA, Nov. 1994, pp. 1298-1303.

Orhan Gazi received the BS and MS degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey in 1996 and 2001. Since 2001, he has been working towards the PhD degree. His research includes error control coding and signal processing. He is currently employed as an Instructor in Cankaya University, where he delivers lectures at the undergraduate level in electrical engineering.

Ali Özgür Yılmaz received the BS degree in electrical engineering from the University of Michigan, Ann Arbor in 1999. He received the MS and PhD degrees at the same school in 2001 and 2003. He has been an Assistant Professor with Middle East Technical University, Turkey since September 2003. His research interests include high rate mobile communication systems, cooperative communications, and radar signal processing.