Block-Based Precoding for Serially Concatenated Codes

Robert G. Maunder, *Member, IEEE,* and Lajos Hanzo, *Fellow, IEEE*

***Abstract*—Precoders have been shown to facilitate iterative decoding convergence towards the Maximum Likelihood (ML) performance in serially concatenated schemes. In this letter, we propose a novel block-based precoder as an alternative to classic convolutional precoders. Furthermore, we demonstrate that the proposed block-based precoder facilitates operation at significantly reduced channel Signal to Noise Ratios (SNRs) in practical schemes, having limited latencies, as well as limited implementational and computational decoding complexities.**

***Index Terms*—Error correction coding, block codes, convolutional codes, trellis codes, information rates.**

I. Introduction

EXTRINSIC Information Transfer (EXIT) chart [1] analysis has revealed the conditions that must be satisfied for serially concatenated codes to facilitate iterative decoding convergence towards the Maximum Likelihood (ML) performance, when the channel's Signal to Noise Ratio (SNR) is sufficiently high. More specifically, the exchange of extrinsic Logarithmic Likelihood Ratios (LLRs)[1] must take place between at least two decoders having EXIT functions that reach the $(1, 1)$ point in the top right-hand corner of the EXIT chart. Satisfactory decoders are those that (a) generate extrinsic LLRs pertaining to a sequence of bits having legitimate permutations that are separated by Hamming distances of at least two and (b) do so by exploiting this fact [2]. Furthermore, recursive codes are satisfactory, provided they are not employed as outer codes [3]. However, these conditions are typically not satisfied by inner codes such as demodulators, despreaders and detectors. When these are serially concatenated with an outer code having an EXIT function that does reach the $(1, 1)$ point, a recursive convolutional precoder [4] can be introduced between the two concatenated codes in order to facilitate iterative decoding convergence towards the ML performance. However, convolutional precoders operate on the basis of memory-demanding trellises, which decode the entire bit sequence at once and impose a high implementational complexity.

In this letter, we propose an alternative precoder that is block-based rather than convolutional. Since each precoded block can be decoded independently, our approach facilitates a more efficient implementation, which can benefit from parallel processing, pipelining and a reduced memory requirement. Furthermore, *we will demonstrate that our precoder has a significantly lower decoding complexity than even the simplest of convolutional precoders, enabling it to outperform this benchmarker, when the affordable latency and computational complexity are limited.*

The authors are with the School of Electronics and Computer Science, University of Southampton, Hampshire, SO17 1BJ, UK, e-mail: {rm,lh}@ecs.soton.ac.uk.

[1]Note that owing to the sub-optimality of iterative decoders, the extrinsic information typically does not convey ratios of true likelihoods. Despite this, we adopt the term 'LLR' due to its ubiquitous usage in this context.

II. The Algorithm

In the transmitter of Figure 1, the proposed block-based precoder encodes the bit sequence $\mathbf{c}$ provided by the outer interleaver π_o in order to generate an input sequence $\mathbf{e}$ for the inner interleaver π_i, similarly to a convolutional precoder. This is achieved using the repetition and check encoder, which generates the $N_\mathbf{d}$-bit sequence $\mathbf{d}$ of Figure 1 by first decomposing the $N_\mathbf{c}$-bit sequence $\mathbf{c}$ into $(N_\mathbf{d} - 1)/2$ equal-length sub-sequences $\{\mathbf{c}_j\}_{j=1}^{(N_\mathbf{d}-1)/2}$, where $N_\mathbf{d}$ is odd. The bits of $\mathbf{d} = \{d_i\}_{i=1}^{N_\mathbf{d}}$ having an index $i \in \{1, 3, 5, \ldots, N_\mathbf{d}-4, N_\mathbf{d}-2\}$ are obtained as the modulo-2 sum of (a) the preceeding bit d_{i-1} and (b) the bits in the sub-sequence $\mathbf{c}_{(i+1)/2}$, where we employ $d_0 = 0$. Meanwhile, the bits having an index $i \in \{2, 4, 6, \ldots, N_\mathbf{d} - 3, N_\mathbf{d} - 1, N_\mathbf{d}\}$ are set equal to the preceeding bit d_{i-1}. As indicated by the crossed block in Figure 1, the bit sequence $\mathbf{e}$ is obtained by multiplexing the bits of $\mathbf{c}$ and $\mathbf{d}$ in the order $[d_{N_\mathbf{d}-1}, d_{N_\mathbf{d}}, \mathbf{c}_1, d_1, d_2, \mathbf{c}_2, d_3, d_4, \mathbf{c}_3, d_5, d_6, \ldots, \mathbf{c}_{(N_\mathbf{d}-1)/2}, d_{N_\mathbf{d}-2}]$, as exemplified for $N_\mathbf{d} = 9$ in Figure 2. The resultant coding rate is given by $R_\mathrm{rc} = N_\mathbf{c}/(N_\mathbf{c} + N_\mathbf{d})$.

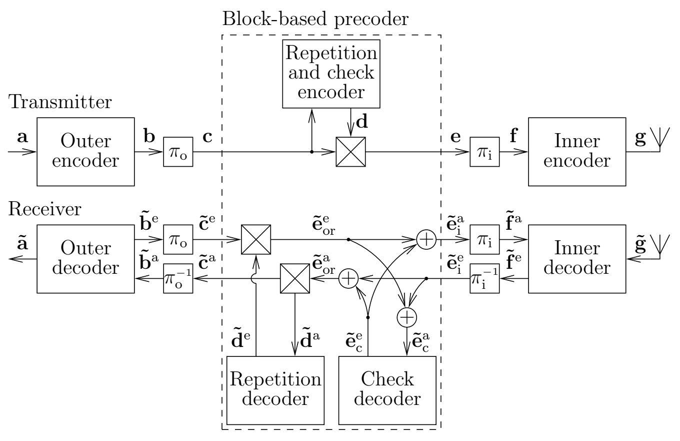
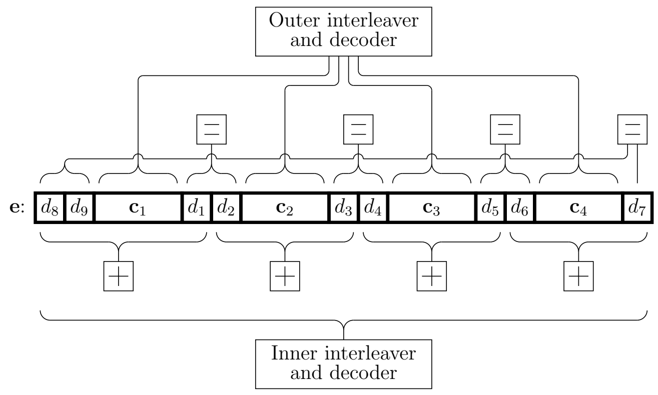

Block-based precoder
Repetition and check encoder
Transmitter
$\mathbf{a}$
Outer encoder
$\mathbf{b}$
π_o
$\mathbf{c}$
$\mathbf{d}$
$\mathbf{e}$
π_i
$\mathbf{f}$
Inner encoder
$\mathbf{g}$
Receiver
$\tilde{\mathbf{a}}$
Outer decoder
$\tilde{\mathbf{b}}^\mathrm{e}$
$\tilde{\mathbf{b}}^\mathrm{a}$
$\tilde{\mathbf{c}}^\mathrm{e}$
$\tilde{\mathbf{c}}^\mathrm{a}$
π_o^{-1}
$\tilde{\mathbf{e}}^\mathrm{e}_\mathrm{or}$
$\tilde{\mathbf{e}}^\mathrm{a}_\mathrm{or}$
$\tilde{\mathbf{e}}^\mathrm{a}_\mathrm{i}$
$\tilde{\mathbf{e}}^\mathrm{e}_\mathrm{i}$
π_i^{-1}
$\tilde{\mathbf{f}}^\mathrm{a}$
$\tilde{\mathbf{f}}^\mathrm{e}$
Inner decoder
$\tilde{\mathbf{g}}$
$\tilde{\mathbf{d}}^\mathrm{e}$
$\tilde{\mathbf{d}}^\mathrm{a}$
$\tilde{\mathbf{e}}^\mathrm{e}_\mathrm{c}$
$\tilde{\mathbf{e}}^\mathrm{a}_\mathrm{c}$
Repetition decoder
Check decoder

Fig. 1. Block-based precoder schematic.

In the receiver, the four decoders of Figure 1 iteratively exchange increasingly more reliable extrinsic LLR sequences



Outer interleaver and decoder
e:
d_8 d_9 $\mathbf{c}_1$ d_1 d_2 $\mathbf{c}_2$ d_3 d_4 $\mathbf{c}_3$ d_5 d_6 $\mathbf{c}_4$ d_7
=
+
Inner interleaver and decoder

Fig. 2. Block-based precoder graph. Braces indicate for which bits each decoder can generate extrinsic LLRs.

$\tilde{\mathbf{e}}^{\mathrm{e}}$, which pertain to the bits of $\mathbf{e}$. The decoders are aided by the *a priori* LLR sequences $\tilde{\mathbf{e}}^{\mathrm{a}}$, which are obtained by summing the extrinsic LLRs provided by the other decoders, as shown in Figure 1. The repetition decoder of Figure 1 generates the extrinsic LLRs $\tilde{\mathbf{d}}^{\mathrm{e}}$ pertaining to the blocks of bits from $\mathbf{d}$ that have identical binary values, as indicated by the boxed equal signs in Figure 2. This decoder exploits the fact that, since each block comprises at least two identical bits, their legitimate permutations are separated by Hamming distances of at least two. As a result, the repetition decoder has an EXIT function that reaches the $(1,1)$ point of the EXIT chart, as described in Section I. The repetition decoder generates the extrinsic LLR pertaining to each bit in a particular block of $\mathbf{d}$ as the sum of the *a priori* LLRs in $\tilde{\mathbf{d}}^{\mathrm{a}}$ that pertain to the *other* bits in the block. Since most blocks contain only two bits, the repetition decoder has a negligible computational complexity.

Similarly, the check decoder of Figure 1 employs the forward-backward algorithm [5] to generate the extrinsic LLRs $\tilde{\mathbf{e}}_{\mathrm{c}}^{\mathrm{e}}$ pertaining to the blocks of bits in $\mathbf{e}$ that are indicated by the boxed plus signs in Figure 2. Owing to the modulo-2 sums employed in the repetition and check encoder, these blocks have even Hamming weights and, hence, legitimate permutations that are separated by Hamming distances of at least two. Since this fact is exploited by the check decoder, its EXIT function also reaches the $(1,1)$ point of the EXIT chart, like that of the repetition decoder. Note that, since the forward-backward algorithm does not consider multiple states, the check decoder typically invokes approximately 50% fewer Add, Compare and Select (ACS) operations than even the simplest of convolutional precoders, which employs two states [4].

Observe in Figure 2 that the combination of the outer and repetition decoders can generate an extrinsic LLR pertaining to every bit in the sequence $\mathbf{e}$. By applying some simple constraints [2], the outer decoder can be designed to have an EXIT function that reaches the $(1,1)$ point, like the repetition decoder. In this case, the combination of these decoders will have a composite EXIT function that also reaches the $(1,1)$ point. Similarly, the check decoder can generate an extrinsic LLR pertaining to every bit in the sequence $\mathbf{e}$ and has an EXIT function that reaches the $(1,1)$ point. Therefore, each extrinsic LLR is exchanged between two decoders having EXIT functions that reach the $(1,1)$ point, even in the case where the inner code's EXIT function does not reach the $(1,1)$ point. As described in Section I, the proposed block-based precoder therefore facilitates iterative decoding convergence towards the ML error ratio performance, as we shall show in Section III.

III. Results

The performance of the scheme shown in Figure 1 was compared to that of two benchmarkers. In the first one, the block-based precoder was replaced by a convolutional precoder, while in the second, the bit sequence $\mathbf{e}$ was provided directly by $\mathbf{c}$, without precoding. In all schemes, the source sequence $\mathbf{a}$ comprised 16-ary source symbols, having values that occur with unequal probabilities, resulting in an entropy of $E = 3.77$ bits/symbol. This motivates the use of an outer Fixed Length Code (FLC), in order to provide joint source and channel coding. Furthermore, in all schemes, the inner code was provided by set partitioned $M_{\mathrm{i}} = 16$-ary Quadrature Amplitude Modulation (16QAM), which has an EXIT function that does not reach the $(1,1)$ point of the EXIT chart. In the proposed 'block-based precoder' scheme, a coding rate of $R_{\mathrm{rc}} = N_{\mathbf{c}}/(N_{\mathbf{c}} + N_{\mathbf{d}}) = 5/6$ was employed for the block-based precoder, while the outer FLC code employed $L_{\mathrm{o}} = 5$-bit codewords, giving a rate of $R_{\mathrm{o}} = E/L_{\mathrm{o}} = 0.754$. By contrast, a two-state convolutional precoder having a coding rate of $R_{\mathrm{p}} = 1$ was employed in the 'convolutional precoder' scheme. In order to maintain a throughput of $\eta = R_{\mathrm{o}}R_{\mathrm{rc}}\log_2(M_{\mathrm{i}}) = 2.51$ bits per channel use and to facilitate fair comparisons, a more error resilient $L_{\mathrm{o}} = 6$-bit FLC having a coding rate of $R_{\mathrm{o}} = 0.628$ was employed in both the 'convolutional precoder' and the 'no precoder' schemes. Note that both the $L_{\mathrm{o}} = 5$- and $L_{\mathrm{o}} = 6$-bit FLCs employed codewords that are separated by Hamming distances of at least two, resulting in FLC EXIT functions that reach the $(1,1)$ point.

We simulated the 'block-based precoder', 'convolutional precoder' and 'no precoder' schemes using uniform interleavers for transmission over an uncorrelated narrowband Rayleigh fading channel having a range of E_b/N_0 values. We elected to consider the transmission of $N_{\mathrm{a}} = 100$-symbol source sequences, since this facilitates the low latency that is required by the challenging low-delay audio, speech and wireless sensor network scenarios. Since the affordable decoding complexity is typically limited in these scenarios, we considered the Symbol Error Ratio (SER) performance that can be achieved without exceeding the complexity limits of 2×10^5, 4×10^5, 8×10^5 and 16×10^5 ACS operations per $N_{\mathrm{a}} = 100$-symbol source sequence. In the 'block-based precoder' scheme, these complexity limits were respectively found to facilitate 3, 5, 10 and 20 iterations of the decoder activation sequence {inner, outer, repetition, check}. Similarly, 2, 4, 7 and 15 iterations of the decoder activation sequence {inner, precoder, outer, precoder} were facilitated in the 'convolutional precoder' scheme, respectively. Finally, the complexity limits respectively facilitated 3, 5, 11 and 22 iterations of the decoder activation sequence {inner, outer} in the 'no precoder' scheme. Our SER performance results are plotted in Figure 3.

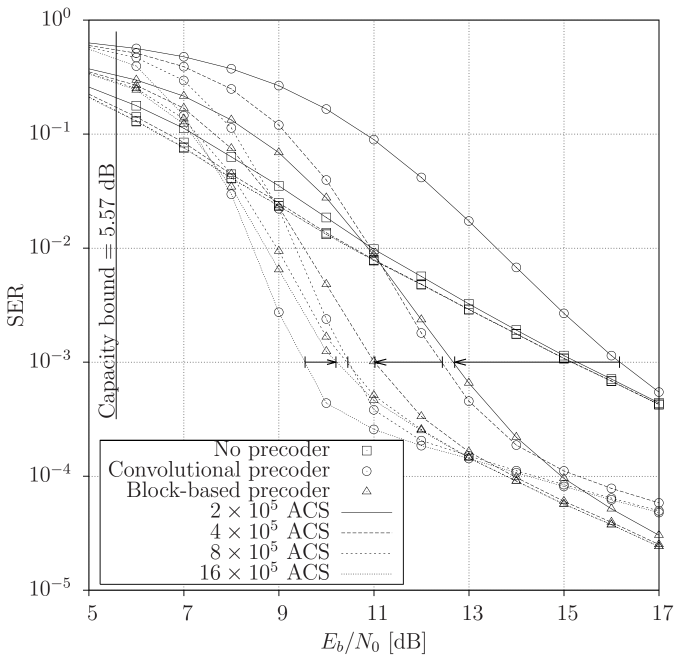


Fig. 3. SER performance of the 'block-based precoder', 'convolutional precoder' and 'no precoder' schemes, for various decoding complexity limits.

Figure 3 shows that, unlike the 'no precoder' scheme, the 'block-based precoder' and 'convolutional precoder' schemes exhibit moderate 'turbo cliffs' and 'error floors'. This demonstrates that, in the scenario considered, precoding is necessary in order to approach the ML SER performance, as described in Section I. As shown in Figure 3, the proposed 'block-based precoder' scheme offers lower SERs than the 'convolutional precoder' scheme in almost all cases considered. In fact, the 'convolutional precoder' scheme is only preferable, when the E_b/N_0 value is within a limited range below 13 dB and when a high decoding complexity in excess of 8×10^5 ACS operations per $N_{\mathbf{a}} = 100$-symbol frame can be afforded. Furthermore, Figure 3 shows that at these high decoding complexities, the iteration gain is limited by a law of diminishing returns. Indeed, we found that none of the schemes considered could offer any substantial iteration gains, when the decoding complexity limit exceeded 16×10^5 ACS operations per $N_{\mathbf{a}} = 100$-symbol frame. Since it is often preferable to curtail iterations before the law of diminishing returns limits the iteration gain, we consider the proposed 'block-based precoder' scheme to be the most desirable arrangement in practical applications.

IV. Conclusions

In this letter, we have proposed a novel block-based precoder, which has a lower implementational complexity than traditional convolutional precoders. More specifically, our block-based precoder facilitates pipelined and parallel processing, since its precoded blocks can be decoded separately. By contrast, convolutional precoders require high-memory trellises, which decode the entire bit sequence at once. Furthermore, we have shown that our block-based precoder also has a lower computational complexity per decoding iteration than even the simplest convolutional precoder. Therefore in practical scenarios, where the affordable latency and decoding complexity are limited, the block-based precoder facilitates more decoding iterations and therefore operation at lower E_b/N_0 values. Indeed, Figure 3 shows that our 'block-based precoder' scheme achieves an SER of 10^{-3} at a 3.5 dB lower E_b/N_0 value than that required by the 'convolutional precoder' scheme, when the affordable decoding complexity is 2×10^5 ACS operations per $N_{\mathbf{a}} = 100$-symbol frame. Similarly, a gain of 1.4 dB was offered, when a decoding complexity of 4×10^5 ACS operations per $N_{\mathbf{a}} = 100$-symbol frame was deemed affordable.

References

[1] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[2] J. Kliewer, N. Görtz, and A. Mertins, "Iterative source-channel decoding with Markov random field source models," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3688–3701, October 2006.

[3] J. Kliewer, A. Huebner, and D. J. Costello, "On the achievable extrinsic information of inner decoders in serial concatenation," in *Proceedings of the IEEE International Symposium on Information Theory*, Seattle, WA, USA, July 2006, pp. 2680–2684.

[4] M. Tüchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 2, Taipei, Taiwan, November 2002, pp. 1358–1362.

[5] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, August 2005.