Adaptive Iterative Decoding for Expediting the Convergence of Unary Error Correction Codes

Wenbo Zhang, Yanbo Jia, Xi Meng, Matthew F. Brejza, Robert G. Maunder, Lajos Hanzo

***Abstract*—Multimedia encoders typically generate symbols having a wide range of legitimate values. In practical mobile wireless scenarios, the transmission of these symbols is required to be bandwidth efficient and error resilient, motivating both source coding and channel coding. However, Separate Source and Channel Coding (SSCC) schemes are typically unable to exploit the residual redundancy in the source symbols, which cannot be totally reduced by finite-delay, finite-complexity schemes, hence resulting in a capacity loss. Until recently, none of the existing Joint Source and Channel Codes (JSCCs) were suitable for this application, since their decoding complexity increases rapidly with the size of the symbol alphabet. Motivated by this, we proposed a novel JSCC referred to as the Unary Error Correction (UEC) code, which is capable of exploiting all residual redundancy and eliminating any capacity loss, while imposing only a moderate decoding complexity. In this paper, we show that the operation of the UEC decoder can be dynamically adapted, in order to strike an attractive trade-off between its decoding complexity and its error correction capability. Furthermore, we conceive the corresponding Three Dimensional (3D) EXtrinsic Information Transfer (EXIT) charts for controlling this dynamic adaptation, as well as the decoder activation order, when the UEC code is serially concatenated with a turbo code. In this way, we expedite the iterative decoding convergence, facilitating a gain of up to 1.2 dB compared to both SSCC and to its non-adaptive UEC benchmarkers, while maintaining the same transmission bandwidth, duration, energy and decoding complexity.**

***Index Terms*—Source coding, Video coding, Channel coding, Channel capacity, Iterative decoding**

I. Introduction

In mobile wireless scenarios, multimedia transmission is required to be bandwidth efficient and resilient to transmission errors, motivating both source and channel coding [1]–[3]. The H.264 video codec [4] employs the Elias Gamma (EG) source code [5] and this may be concatenated with diverse codes, such as a Convolutional Code (CC) to provide separate channel coding, for example. However, this Separate Source and Channel Coding (SSCC) scheme typically suffers from a capacity loss, owing to the residual redundancy, which is retained following EG encoding [6]. This limits both the achievable bandwidth efficiency and its resilience to transmission errors in mobile wireless scenarios.

In many applications, the classic SSCC schemes may be replaced by Joint Source and Channel Codes (JSCCs) [7], in order to exploit the residual redundancy which cannot be completely recoved by the finite-delay, finite-complexity schemes and hence to avoid capacity loss. However, the symbols that are EG encoded in the H.264 video codec are approximately zeta distributed [6, Fig. 1]. More specifically, while most of these symbols have low values, some can have very high values of around 1000. Until recently, the large cardinality of this symbol value set prevented the application of all existing JSCCs. This is because the decoding complexity of these JSCCs increases rapidly with the cardinality of the symbol set, asymptotically becoming infinite when the cardinality is infinite [6]. Motivated by this, we proposed the novel Unary Error Correction (UEC) code family [6], which is the only JSCC that mitigates capacity loss and maintains a moderate decoding complexity, even when the cardinality of the symbol set is infinite. In [6], we demonstrated that an iteratively-decoded serial concatenation of the UEC code with an Irregular Unity Rate Code (IrURC) is capable of providing a 1.3 dB gain compared to a SSCC benchmarker employing separate EG and CC codes, while maintaining the same transmission bandwidth, throughput, latency, energy dissipation and decoding complexity.

Against this background, this paper proposes an adaptive iterative decoding technique for expediting the iterative decoding convergence of UEC codes, facilitating an approximately 1.2 dB gain compared to a benchmarker employing the non-adaptive UEC scheme of [6]. We commence in Section II by summarising the operation of the UEC code, in the new context where it is concatenated with a turbo code. Both the UEC encoder and decoder operate on the basis of trellises, in which the transitions between the states are synchronous with the transitions between consecutive codewords in a unary-encoded bit sequence. The UEC decoder applies the classic Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [8] to its trellis, for the sake of exploiting the residual redundancy in the unary-encoded bit sequence and hence for mitigating the potential capacity loss. However, we show for the first time that the number of states employed by the UEC decoder can be adapted independently of the number of states employed by the UEC encoder, in order to strike a trade-off between the attainable error correction capability and the decoding complexity imposed.

This idea is extended in Section III, allowing not only the dynamic adjustment of the UEC decoder's operation, but also the dynamic adjustment of its activation order with the two turbo decoder components. We propose the employment of Three Dimensional (3D) EXtrinsic Information Transfer (EXIT) charts [9] for quantifying the benefit of activating each

The authors are with Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom, e-mail: {wz4g11,mfb2g09,rm,lh}@ecs.soton.ac.uk

The financial support of the EPSRC, Swindon UK under the grant EP/J015520/1, that of the RCUK under the India-UK Advanced Technology Centre (IU-ARC) as well as of the EU under the CONCERTO project and that of the European Research Council's Advanced Fellow grant is gratefully acknowledged.

decoding component, at each stage of the iterative decoding process. At the same time, we quantify the corresponding cost in terms of the computational complexity of each decoding component. Moreover, the 3D EXIT chart can be projected into two dimensions [10], [11], which offers insights into whether or not any capacity loss is expected for the scheme. By activating the specific decoding component offering the largest benefit-to-cost ratio at each stage, we demonstrate that the convergence of the iterative decoding process may be significantly expedited. Furthermore, we demonstrate that the storage required for implementing the proposed adaptive iterative decoding scheme is modest compared to the storage required by the interleavers, for example.

In Section IV, both the capacity loss and the Symbol Error Ratio (SER) performance of the proposed Adaptive UEC-Turbo scheme is compared to those of the benchmarkers employing either a SSCC or the non-adaptive UEC scheme of [6]. Our results show that while operating within 3.6 dB of the capacity bound, the proposed scheme offers as much as 1.2 dB gain over the best benchmarker, while maintaining the same transmission bandwidth, throughput, latency, energy dissipation and decoding complexity. Finally, Section V concludes the paper.

II. Transmitter and Receiver Operation

In this section, we detail the operation of the UEC-Turbo scheme of Fig. 1. The transmitter's operation is described in Section II-A, while the receiver is considered in Section II-B, where we show that the number of states employed by the UEC decoder can be adapted independently of the number of states employed by the UEC encoder.

A. Transmitter

The UEC encoder of [6] is designed for conveying a vector $\mathbf{x} = [x_i]_{i=1}^{a}$ comprising a number of symbols, as shown in Fig. 1. The value of each symbol $x_i \in \mathbb{N}_1$ may be modeled by an Independent and Identically Distributed (IID) Random Variable (RV) X_i, which adopts the value x with a probability of $\Pr(X_i = x) = P(x)$, where $\mathbb{N}_1 = \{1, 2, 3, \ldots\}$ is the infinite-cardinality set comprising all positive integers. Throughout this paper we assume that the symbol values obey a zeta probability distribution [12], since this is typical of the symbols produced by multimedia encoders, as described in Section I. The zeta probability distribution is defined as

$$P(x) = \frac{x^{-s}}{\zeta(s)}, \tag{1}$$

where $\zeta(s) = \sum_{x\in\mathbb{N}_1} x^{-s}$ is the Riemann zeta function, $s > 1$ parametrizes the zeta distribution $P(x) = \frac{x^{-s}}{\zeta(s)}$, where $-s$ is the gradient of the zeta distribution when $P(x)$ is plotted against x using a log-log axis, as shown in [6, Fig. 1]. Alternatively, the zeta distribution may be parameterized by $p_1 = \Pr(X_i = 1) = 1/\zeta(s)$, which is the occurrence probability of the most frequently encountered symbols, namely those having a value of 1. More p_1 values have been investigated in [13]. In the situation, where the symbols obey the zeta distribution of (1), the symbol entropy is given by

$$H_X = \sum_{x\in\mathbb{N}_1} H[P(x)] = \frac{\ln(\zeta(s))}{\ln(2)} - \frac{s\zeta'(s)}{\ln(2)\zeta(s)}, \tag{2}$$

where $H[p] = p\log_2(1/p)$ and $\zeta'(s) = -\sum_{x\in\mathbb{N}_1} \ln(x)x^{-s}$ is the derivative of the Riemann zeta function.

TABLE I: The first ten symbol probabilities for a zeta distribution having $p_1 = 0.797$, as well as the corresponding unary and EG codewords.

x_i	$P(x_i)$	$\mathbf{y}_i$	
		Unary	EG
1	0.797	0	1
2	0.117	10	010
3	0.038	110	011
4	0.017	1110	00100
5	0.009	11110	00101
6	0.006	111110	00110
7	0.004	1111110	00111
8	0.003	11111110	0001000
9	0.002	111111110	0001001
10	0.001	1111111110	0001010

As shown in Fig. 1, the UEC encoder represents the source vector $\mathbf{x}$ using a unary encoder. More specifically, each symbol x_i in the vector $\mathbf{x}$ is represented by a corresponding codeword $\mathbf{y}_i$ that comprises x_i bits, as exemplified in Table I. When the symbols adopt the zeta distribution of (1), the average unary codeword length l is only finite for $s > 2$ and hence for $p_1 > 0.608$ [6], in which case we have

$$l = \sum_{x\in\mathbb{N}_1} P(x)\cdot x = \frac{\zeta(s-1)}{\zeta(s)}. \tag{3}$$

Note that for $p_1 \leq 0.608$, our Elias Gamma Error Correction (EGEC) code of [13] may be employed in order to achieve a finite average codeword length. In the scenario, where $p_1 = 0.797$, an average codeword length of $l = 1.54$ results. The output of the unary encoder is generated by concatenating the selected codewords $[\mathbf{y}_i]_{i=1}^{a}$, in order to form the b-bit vector $\mathbf{y} = [y_j]_{j=1}^{b}$. For example, the source vector $\mathbf{x} = [1, 1, 5, 1, 1, 2, 3, 1, 1, 6]$ of $a = 10$ symbols yields the $b = 22$-bit vector $\mathbf{y} = [0011110001011000111110]$. Note that the average length of the bit vector $\mathbf{y}$ is given by $(a \cdot l)$.

Following unary encoding, the UEC encoder employs a trellis to encode the bit sequence $\mathbf{y}$, as shown in Fig. 1. The generalized UEC trellis of [6, Fig. 3(a)] is parametrized by the codebook $\mathbb{C}$, which comprises $r/2$ number of n-bit codewords, where r is the number of trellis states employed. For example, the codebook $\mathbb{C} = \{1\}$ corresponds to the $r = 2$-state $n = 1$-bit UEC trellis of Fig. 2, while $\mathbb{C} = \{1, 1, 1\}$ yields the $r = 6$-state $n = 1$-bit UEC trellis of Fig. 3, respectively.

Each bit y_j of the input bit sequence $\mathbf{y} = [y_j]_{j=1}^{b}$ forces the trellis encoder to traverse from its previous state $m_{j-1} \in \{1, 2, \ldots, r\}$ to its next state $m_j \in \{1, 2, \ldots, r\}$, in order of increasing bit-index j. Each next state m_j is selected from two legitimate alternatives, depending on the bit value y_j, according to

$$m_j = \begin{cases} 1 + \mathrm{odd}(m_{j-1}) & \text{if } y_j = 0 \\ \min[m_{j-1} + 2, r - \mathrm{odd}(m_{j-1})] & \text{if } y_j = 1 \end{cases}, \tag{4}$$

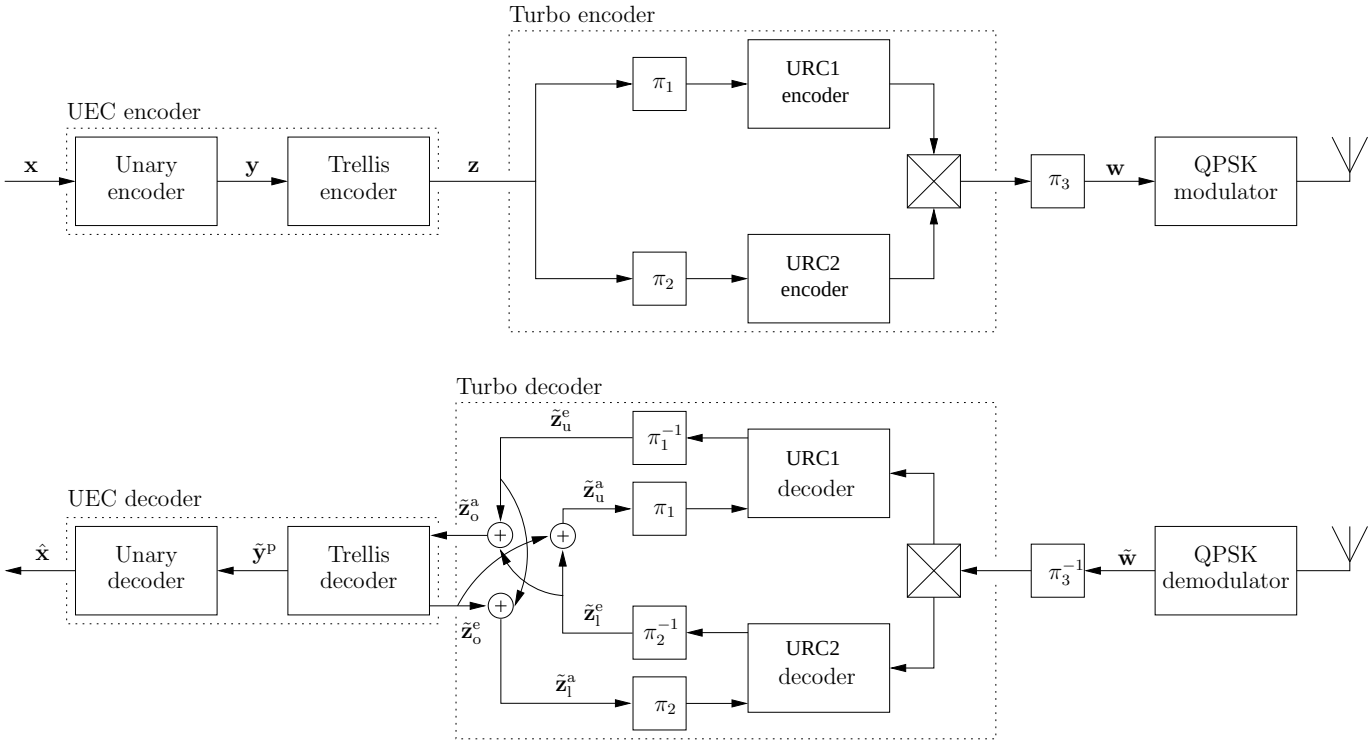

Fig. 1: Schematic of the UEC-Turbo scheme, in which a UEC code is serially concatenated with turbo coding and Gray-coded QPSK modulation schemes. Bold notation without a diacritic is used to denote a symbol vector or a bit vector. A diacritical hat represents a reconstruction of the symbol or bit vector having the corresponding notation. A diacritical tilde represents an LLR vector pertaining to the bit vector with the corresponding notation. The superscripts 'a', 'e' and 'p' denote *a priori*, extrinsic and *a posteriori* LLR vectors, respectively. The subscripts 'o', 'u' and 'l' denote relevance to the outer UEC code, the upper turbo component code and the lower turbo component code, respectively. Here, π_1, π_2 and π_3 represent interleavers, while π_1^{-1}, π_2^{-1} and π_3^{-1} represent the corresponding deinterleavers. Multiplexer and de-multiplexer operations are also employed before π_3 and after π_3^{-1}, respectively.

where the number of possible states r is required to be even and the encoding process always emerges from the state $m_0 = 1$. The function $\mathrm{odd}(\cdot)$ yields 1 if the operand is odd or 0 if it is even. In this way, the bit vector $\mathbf{y}$ identifies a path through the trellis, which may be represented by a vector $\mathbf{m} = [m]_{j=0}^{b}$ comprising $(b+1)$ state values. Note that the example bit vector $\mathbf{y}$ provided above yields the path $\mathbf{m} = [1,2,1,1,1,1,1,2,1,2,2,1,1,1,2,1,2,2,2,2,2,2,1]$ through the $r = 2$-state trellis of Fig. 2. Similiarly, the same bit vector $\mathbf{y}$ yields the path $\mathbf{m} = [1,2,1,3,5,5,5,2,1,2,4,1,3,5,2,1,2,4,6,6,6,6,1]$ through the $r = 6$-state trellis of Fig. 3. Note that the trellis path $\mathbf{m}$ is guaranteed to terminate in the state $m_b = 1$, when the symbol vector $\mathbf{x}$ has an even length a, while $m_b = 2$ is guaranteed, when a is odd [6].

The path vector $\mathbf{m}$ may be modeled as a particular realization of a vector $\mathbf{M} = [M_j]_{j=0}^{b}$ comprising $(b+1)$ RVs, which are associated with the joint transition probabilities $\Pr(M_j = m, M_{j-1} = m') = P(m,m')$ of [6, Equation (8)] and the conditional transition probabilities $\Pr(M_j = m | M_{j-1} = m') = P(m|m')$ of [14, Equation (6)].

The UEC trellis encoder represents each bit y_j in the vector $\mathbf{y}$ by an n-bit codeword $\mathbf{z}_j$. This is selected from the codebook $\mathbb{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{r/2-1}, \mathbf{c}_{r/2}\}$ or from the complementary codebook $\overline{\mathbb{C}} = \{\overline{\mathbf{c}_1}, \overline{\mathbf{c}_2}, \ldots, \overline{\mathbf{c}_{r/2-1}}, \overline{\mathbf{c}_{r/2}}\}$, which is achieved according to

$$\mathbf{z}_j = \begin{cases} \overline{\mathbf{c}_{\lceil m_{j-1}/2 \rceil}} & \text{if } y_j = \mathrm{odd}(m_{j-1}) \\ \mathbf{c}_{\lceil m_{j-1}/2 \rceil} & \text{if } y_j \neq \mathrm{odd}(m_{j-1}) \end{cases} . \quad (5)$$

Then, the selected codewords are concatenated to obtain the bn-bit vector $\mathbf{z} = [z_k]_{k=1}^{bn}$ of Fig. 1. Recall that the $r = 2$- and $r = 6$-state trellises of Fig. 2 and Fig.3 yield the different paths $\mathbf{m}$, when encoding the same example bit vector $\mathbf{y}$ provided above. Despite this however, both result in the same encoded bit vector $\mathbf{z} = [10000101100010111110]$.

Note that the UEC-encoded bit vector $\mathbf{z}$ will always be

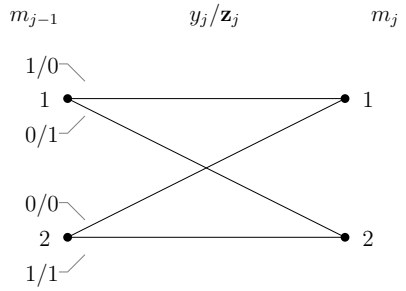

Fig. 2: An $r = 2$-state $n = 1$-bit UEC trellis, where $\mathbb{C} = \{1\}$.

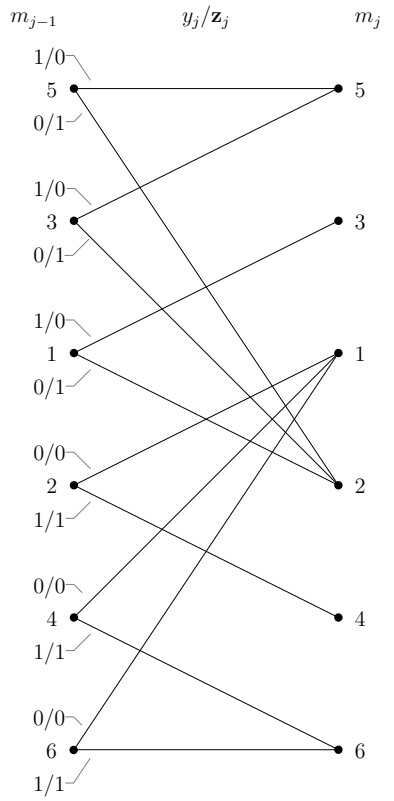

Fig. 3: An $r = 6$-state $n = 1$-bit UEC trellis, where $\mathbb{C} = \{1, 1, 1\}$.

identical, regardless of whether the $r = 2$-state UEC trellis of Fig. 2 or the $r = 6$-state UEC trellis of Fig. 3 is employed. This is because the $r = 6$ codebook $\mathbb{C} = \{1, 1, 1\}$ may be considered to be an extension of the $r = 2$ codebook $\mathbb{C} = \{1\}$. More specifically, a codebook $\mathbb{C}$ may be extended by appending replicas of the final codeword in the codebook. For example, the $r = 2$ codebook $\mathbb{C} = \{1\}$ can also be extended both to the $r = 4$ codebook $\mathbb{C} = \{1, 1\}$ and to the $r = 8$ codebook $\mathbb{C} = \{1, 1, 1, 1\}$. In fact, regardless of how much a codebook $\mathbb{C}$ is extended, the UEC-encoded bit vector $\mathbf{z}$ will be identical, when encoding the same symbol vector $\mathbf{x}$. We will exploit this property in Section III for dynamically adjusting the number of states employed in the UEC decoder.

The bit vector $\mathbf{z}$ may be modeled as a particular realization of a vector $\mathbf{Z} = [Z_k]_{k=1}^{bn}$ comprising bn binary RVs. Each binary RV Z_k adopts the values 0 and 1 with the probabilities $\Pr(Z_k = 0)$ and $\Pr(Z_k = 1)$ respectively, corresponding to a bit entropy of $H_{Z_k} = H[\Pr(Z_k = 0)] + H[\Pr(Z_k = 1)]$. The overall average coding rate R_o of the UEC encoder is given by

$$R_\mathrm{o} = \frac{aH_X}{\sum_{k=1}^{bn} H_{Z_k}}. \tag{6}$$

Note that the average coding rate of the UEC encoder depends upon the parameter p_1 of the zeta distribution, as shown in [6, Fig. 5]. Furthermore, the UEC trellis is designed to be symmetric and to rely on complementary codewords, so that it produces equiprobable bit values, where $\Pr(Z_k = 0) = \Pr(Z_k = 1) = 0.5$ and giving a bit entropy of $H_{Z_k} = 1$. In this case, (6) reduces to $R_\mathrm{o} = H_X/(ln)$, which is identical to [6, Equation (11)]. When the source symbols obey a zeta distribution having $p_1 = 0.797$, we obtain a UEC coding rate of $R_\mathrm{o} = 0.762$, as shown in Table II, which is in agreement with [6, Fig. 5].

As shown in Fig. 1, the UEC-encoded bit vector $\mathbf{z}$ is then forwarded to the turbo encoder. This encodes the bit vector $\mathbf{z}$ twice, employing the pair of interleavers π_1 and π_2 to make the two encoded data sequences approximately statistically independent of each other. Here, the pair of turbo component encoders are constituted by identical $r = 8$-state Unity Rate Code (URC) encoders, which are labeled URC1 and URC2 in Fig. 1. These employ $(8, \mathrm{F}) = [1001, 1111]$ as the octally represented feedforward and feedback polynomials respectively, as depicted in the eighth schematic of [15, Fig. 9.6]. The coding rate R_i of the inner turbo code is given by

$$R_\mathrm{i} = \frac{\sum_{k=1}^{bn} H_{Z_k}}{2bn}, \tag{7}$$

where the coding rate of each individual URC encoder is equal to $2R_\mathrm{i}$. In the scenario, where $H_{Z_k} = 1$, we obtain $R_\mathrm{i} = 1/2$, as shown in Table II and as discussed above.

After multiplexing the resultant encoded bit sequences, the channel interleaver π_3 is employed in Fig. 1 for the sake of dispersing burst errors. Following this, Gray-coded $M = 4$-ary Quaternary Phase Shift Keying (QPSK) modulation may be employed for transmitting the resultant bit vector $\mathbf{w}$, as shown in Fig. 1. The effective throughput of the UEC-Turbo scheme is given by $\eta = R_\mathrm{o} \cdot R_\mathrm{i} \cdot \log_2(M)$, which has the value of $\eta = 0.762$ information bits/symbol, when the source symbols obey a zeta distribution having $p_1 = 0.797$, as shown in Table II. Alternatively, a mapping scheme other than Gray coding or a modulation scheme having a higher order M can be employed, although this may require a higher complexity receiver design [6].

B. Receiver

In the receiver of Fig. 1, QPSK demodulation is employed in order to obtain the LLR vector $\tilde{\mathbf{w}}$. This is deinterleaved π_1^{-3} and demultiplexed, before iterative decoding commences by exchanging extrinsic information among the URC1, URC2 and UEC decoders of Fig. 1. Note that higher-order modulation schemes may be readily employed, although this would require the iterative exchange of extrinsic information between the demodulator and the URC decoders, in order to avoid capacity loss [13]. Our future work will consider adaptive iterative decoding techniques for this four-stage concatenation.

In the three-state concatenation of Fig. 1, the interleavers π_1, π_1^{-1}, π_2 and π_2^{-2} are employed for mitigating the correlation within the *a priori* Logarithmic Likelihood Ratio (LLR) vectors of $\tilde{\mathbf{z}}_\mathrm{o}^\mathrm{a}$, $\tilde{\mathbf{z}}_\mathrm{u}^\mathrm{a}$ and $\tilde{\mathbf{z}}_\mathrm{l}^\mathrm{a}$. All three decoders apply the BCJR algorithm [8], which has a complexity directly dependent on the number of states employed. We assume perfect synchronization between the UEC trellis and the unary codewords during the BCJR algorithm's γ_t calculation of [8, Equation (9)]. This employs the conditional transition probability $\Pr(M_j = m | M_{j-1} = m')$ of [14, Equation (6)]. During

TABLE II: Characteristics of the various schemes considered. $E_{\rm b}/N_0$ bounds are given for the case of uncorrelated narrowband Rayleigh fading channel.

Scheme	r	$R_{\rm o}$	$A_{\rm o}$	$R_{\rm i}$	η	$E_{\rm b}/N_0$ [dB] capacity bound	$E_{\rm b}/N_0$ [dB] area bound	$E_{\rm b}/N_0$ [dB] tunnel bound
UEC-Turbo	2	0.762	0.934	0.500	0.762	0.84	2.48	2.7
	4		0.808				1.27	1.8
	6		0.783				1.04	1.3
	8		0.774				0.95	1.3
EG-URC-Turbo	2	0.762	0.882	0.500			1.27	2.4
Unary-Turbo	N/A	0.816	N/A	0.467			2.48	3.1
EG-Turbo	N/A	0.864	N/A	0.441			2.18	2.8

BCJR decoding, the UEC trellis should emerge from $m_0 = 1$ and be terminated either at $m_b = 1$ or $m_b = 2$, depending on whether the length a of the symbol vector $\mathbf{x}$ is even or odd, respectively. Note that since extending a UEC trellis does not change the UEC-encoded bit vector $\mathbf{z}$, the UEC decoder may decode it using an extended, larger-complexity version of the specific UEC trellis employed by the UEC encoder. Increasing the number of states r employed by the UEC trellis decoder in this way has the benefit of improving its error correction capability, at the cost of increasing its complexity [6]. In Section III, we will exploit this to dynamically adjust r for the sake of striking an attractive the trade-off between its decoding complexity and error correction capability.

Observe in Fig. 1 that the *a priori* LLR vectors provided for each of the three decoders are obtained as the sum of the extrinsic LLR vectors most recently generated by the other two decoders, namely we have $\tilde{\mathbf{z}}_{\rm o}^{\rm a} = \tilde{\mathbf{z}}_{\rm u}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}$, $\tilde{\mathbf{z}}_{\rm u}^{\rm a} = \tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}$ and $\tilde{\mathbf{z}}_{\rm l}^{\rm a} = \tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm u}^{\rm e}$. These LLR vectors comprise b number of LLRs, which pertain to the corresponding bits of $\mathbf{z}$. At the start of the iterative decoding process, the extrinsic LLR vectors $\tilde{\mathbf{z}}_{\rm o}^{\rm e}$, $\tilde{\mathbf{z}}_{\rm u}^{\rm e}$ and $\tilde{\mathbf{z}}_{\rm l}^{\rm e}$ are initialized with zero-valued LLRs.

During the iterative decoding process, the iterative operation of the URC1, URC2 and UEC decoders of Fig. 1 may be performed using a wide variety of different decoder activation orderings. For the sake of conceptional simplicity, a fixed decoder activation order may be employed, in which the URC1, URC2 and UEC decoders of Fig. 1 are activated using a regular activation order repeated periodically. For example, the decoders may be consecutively operated in turn, according to $\{$URC1, URC2, UEC; URC1, URC2, UEC; ...$\}$, where each ordered consecutive operation of the URC1, URC2 and UEC decoders represents a full system iteration of the decoding process. Alternatively, we may employ a non-periodic decoder activation order, in which an on-line decision is made at each stage of the iterative decoding process, in order to adaptively and dynamically select which of the URC1, URC2 and UEC decoders of Fig. 1 to activate next. This is exploited in Section III using a novel 3D EXIT chart aided technique, in order to expedite iterative decoding convergence towards an approximation of the Maximum Likelihood (ML) error correction performance.

Following the achievement of iterative decoding convergence, the UEC trellis decoder of Fig. 1 may invoke the BCJR algorithm for generating the vector of *a posteriori* LLRs $\tilde{\mathbf{y}}^{\rm p}$ that pertain to the corresponding unary-encoded bits in the vector $\mathbf{y}$. Then, the unary decoder sorts the vector of *a posteriori* LLRs $\tilde{\mathbf{y}}^{\rm p}$, in order to identify the a number of bits in the vector $\mathbf{y}$ that are most likely to have values of zero. A hard decision bit vector $\hat{\mathbf{y}}$ is then obtained by setting the value of these bits to zero and the value of all other bits to one. Finally, the bit vector $\hat{\mathbf{y}}$ can be unary decoded in order to obtain the symbol vector $\hat{\mathbf{x}}$, which is guaranteed to comprise a number of symbols.

III. Adaptive Iterative Decoding

In this section, we propose our novel adaptive iterative decoding technique for the UEC-Turbo scheme of Fig. 1. As described in Section I, this evaluates the benefit and cost associated with activating each of the URC1, URC2 and UEC trellis decoders of Fig. 1 at each stage of the iterative decoding process, in order to decide as to which decoder to activate next. In the case of the UEC trellis decoder, the number of states r to employ is also considered, as described in Section II.

In Section III-A, we employ 3D EXIT chart analysis [16], [17] in order to quantify the error correction benefit [18] that is offered by each decoder, if activated for the next decoding operation. Meanwhile, the computational complexity cost of each option is quantified in terms of the associated Add, Compare and Select (ACS) arithmetic operations in Section III-B. Hence, by jointly considering the benefit and cost, we can dynamically adapt the iterative decoding process of the UEC-Turbo scheme of Fig. 1, as described in Section III-C. Finally, Section III-D shows that the storage requirements of the proposed technique are modest, compared to those of the interleavers of Fig. 1.

A. EXIT chart analysis

In this section, we analyze the iterative decoding convergence of the UEC-Turbo scheme introduced in Fig. 1, which serially concatenates the UEC code [6] with a turbo code. In Section III-A1, we consider the 2D EXIT curves of the URC1, URC2 and UEC trellis decoders of Fig. 1 separately. Then Section III-A2 shows that the three 2D EXIT curves can be converted into 3D surfaces and plotted in the same 3D EXIT chart, allowing the explicit visualization of the iterative decoding trajectory, when employing any arbitrary decoder activation order. Furthermore, the 3D EXIT chart may be employed for quantifying the benefit, offered by each decoder in terms of the achievable MI improvement. Finally, Section III-A3 shows that the 3D EXIT chart may be projected into two dimensions, in order to demonstrate that the proposed UEC-Turbo scheme facilitates near-capacity operation.

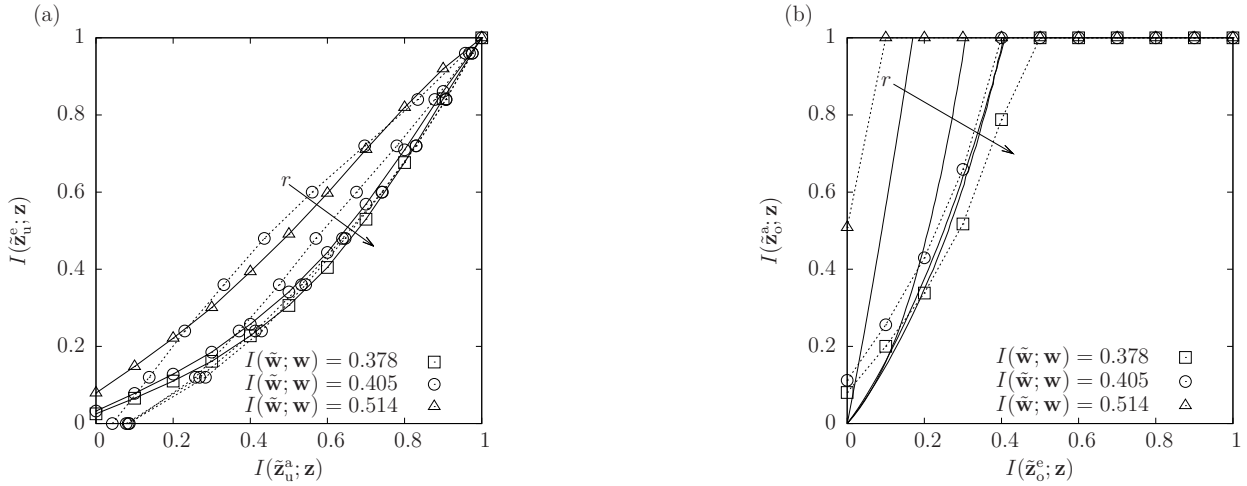

Fig. 4: 2D EXIT charts for the UEC-Turbo scheme, for UEC trellises having various numbers of states $r \in \{2, 4, 6, 8\}$. These EXIT charts are projected in terms of (a) the URC1 decoder and (b) the UEC trellis decoder. In (a), the solid lines represent the URC1 EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{u}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ for a variety of MI values $I(\tilde{\mathbf{w}};\mathbf{w})$. Meanwhile the dotted lines represent the projected UEC-URC2 EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}) = f_{\mathrm{o,l}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w}), r]$, for the case where $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.405$. In (b), the solid lines denote the inverted UEC EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}), r]$, while the dotted lines denote the projected URC1-URC2 EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = f_{\mathrm{u,l}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$. Here, MI values of $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.378, 0.405$ and 0.514 correspond to uncorrelated narrowband Rayleigh fading channel E_{b}/N_0 values of 0.8 dB, 1.3 dB and 3.3 dB, respectively. Different random designs are employed for the interleavers π_1, π_2 and π_3 in each simulated frame.

1) 2D EXIT curves: As shown in Fig. 1, the URC1 decoder generates the extrinsic LLR vector $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}$ with the aid of the *a priori* LLR vector $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}}$, which is combined with the LLRs received over the channel. Therefore, the MI $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z})$ of $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}$ is related to the MI $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z})$ of $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}}$, as well as to the MI $I(\tilde{\mathbf{w}};\mathbf{w})$ of the LLRs $\tilde{\mathbf{w}}$ provided by the demodulator, which depends on the channel's Signal to Noise Ratio (SNR) per bit, namely on $E_{\mathrm{b}}/N_0 = \mathrm{SNR}/\eta$. As detailed in [19], this relationship may be visualized using the 2D EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{u}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ of the URC1 decoder, which is plotted for various $I(\tilde{\mathbf{w}};\mathbf{w})$ values in the 2D EXIT chart of Fig. 4(a). Note that because the URC1 decoder is recursive [20], and hence has an infinite impulse response, it may be referred to as Maximum Mutual Information Achieving (MMIA) [16]. As a result of this, the URC1 EXIT function reaches the $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = 1$ point of pefect decoding convergence associated with the top-right corner of the EXIT chart of Fig. 4(a), where low decoding error rates are facilitated [20]. More explicitly, a low decoding error rate is achieved, because in the presence of perfect *a priori* information, perfect extrinsic information is generated, again, as represented by reaching the $(1, 1)$ point of Fig. 4(a). Likewise, since URC2 of Fig. 1 is identical to URC1, its EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{l}}[I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{a}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ is identical to that of URC1, therefore also reaching the $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = 1$ point.

Similarly, the UEC decoder's transformation of the *a priori* LLR vector $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}}$ into the extrinsic LLR vector $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}$ may be characterized by the UEC EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}), r]$. This EXIT function is shown inverted - i.e. with the abscissa and ordinate axes swapped - in the 2D EXIT chart of Fig. 4(b) for the cases of employing the UEC codebooks $\mathbb{C} = \{1\}$, $\mathbb{C} = \{1, 1\}$, $\mathbb{C} = \{1, 1, 1\}$ and $\mathbb{C} = \{1, 1, 1, 1\}$, which correspond to $r = 2$, 4, 6 and 8 UEC trellis states, respectively. Note that when employing these $n = 1$-bit UEC codebooks, a free distance of $d_{\mathrm{free}} = 1$ results for the UEC-encoded bit vector $\mathbf{z}$. Note that because we have $d_{\mathrm{free}} < 2$, these UEC codebooks are not MMIA [21]. Owing to this distance-limitation, the inverted UEC EXIT functions of Fig. 4(b) do not reach the $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = 1$ point, where low decoding error rates are facilitated. Nevertheless, this does not prevent iterative decoding convergence towards a low decoding error rate, since the presence of the two MMIA URC1 and URC2 decoders is sufficient for achieving this [16]. Note that inverted EXIT curves corresponding to a selection of $n = 2$-bit UEC codebooks can be seen in [6, Fig. 6].

2) 3D EXIT chart: Observe in Fig. 1 that the receiver employs an iterative exchange of LLRs that pertain to the bit sequence $\mathbf{z}$ among three decoding components, i.e., the URC1 decoder, the URC2 decoder and the UEC's trellis decoder of Fig. 1. The *a priori* LLR sequence $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}}$ is obtained as a sum of the extrinsic LLR sequences generated by the other two components $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}} = \tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}} + \tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}$. Hence, the inverted EXIT function of the UEC component, which is denoted by $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}), r]$ may be expressed as a 3D function of two arguments, i.e. as $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}};\mathbf{z}), r]$. Note that for the scenario, where the extrinsic LLRs in the sequences $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}$ and $\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}$ are Gaussian distributed, the MI of the *a priori* LLR sequence $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}}$ can be obtained analytically, according to [16], [22]

$$I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = J\left(\sqrt{J^{-1}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z})]^2 + J^{-1}[I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}};\mathbf{z})]^2}\right), \quad (8)$$

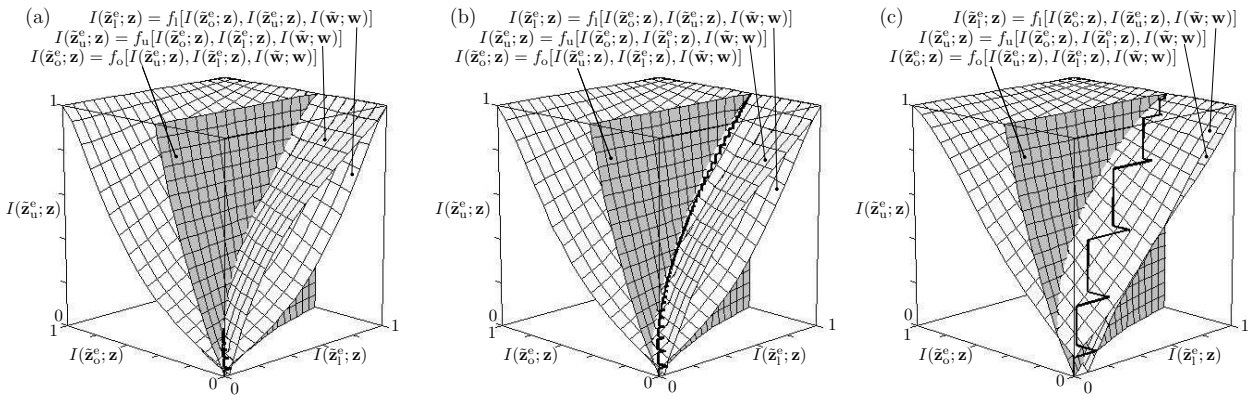

Fig. 5: 3D EXIT surfaces for the UEC-Turbo scheme when employing $r = 6$ states in the UEC trellis decoder, for communicating over an uncorrelated narrowband Rayleigh fading channel, having MI values of (a) $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.378$, (b) 0.405, and (c) 0.514, which correspond to uncorrelated Rayleigh fading channel $E_{\rm b}/N_0$ values of 0.8 dB, 1.3 dB and 3.3 dB, respectively. In each case, a bold line is used to indicate the iterative decoding trajectory, when employing the fixed decoder activation order {URC1, URC2, UEC; URC1, URC2, UEC; ...}. Note that these iterative decoding trajectories reside *below* the URC1 EXIT surface and *above* the two other EXIT surfaces. Different random designs are employed for the interleavers π_1, π_2 and π_3 in each simulated frame.

where

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{e^{-(\xi-\sigma^2/2)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} \cdot \log_2[1+e^{-\xi}]d\xi$$

and $\sigma = J^{-1}(I)$ is the inverse function. In practice, the approximations of [22] may be used for these functions. This approach allows the 3D EXIT function $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}) = f_{\rm o}[I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}), r]$ to be obtained from the 2D EXIT function $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}) = f_{\rm o}[I(\tilde{\mathbf{z}}_{\rm o}^{\rm a};\mathbf{z}), r]$, without requiring time-consuming Monte-Carlo simulations.

In a similar manner, the URC1 component's EXIT function of $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}) = f_{\rm u}[I(\tilde{\mathbf{z}}_{\rm u}^{\rm a};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ seen in Fig. 4(a) may be expressed as $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}) = f_{\rm u}[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$. Note that since the URC1 and URC2 components of the turbo code shown in Fig. 1 are identical, we may directly obtain the URC2 component's 3D function $I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}) = f_{\rm l}[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$, where we have $f_{\rm u} = f_{\rm l}$. Since all three of the 3D EXIT functions $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z})$, $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z})$ and $I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z})$ have the same arguments, they can all be plotted in a single 3D EXIT chart having axes labelled with these three MIs. This approach is shown in Fig. 5 for our UEC-Turbo scheme of Fig. 1, when communicating over uncorrelated narrowband Rayleigh fading channels having a range of $I(\tilde{\mathbf{w}};\mathbf{w})$ values. Here, the complex channel gain has a mean of 0, a variance of 1 and a coherence time of 0, hence each fading coefficient affects only a single transmitted symbol. Note that we employ this channel model, since it is representative of various wireless channels. More specifically, the presence of the interleaver π_3 means that the EXIT functions of URC1 and URC2 remain unaffected by the coherence time of the Rayleigh fading channel, provided that it is short compared to the transmit duration of each frame. Owing to this, our results apply not only to uncorrelated Rayleigh fading channels, but also to fast fading channels.

For example, as shown in Fig. 5(b), the surfaces $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}) = f_{\rm u}[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ and $I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}) = f_{\rm l}[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ - where we have $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.405$ - represent the extrinsic information produced by the URC1 and URC2 components of the turbo decoder for $E_{\rm b}/N_0 = 1.3$ dB. Note that the intersection between these two surfaces and the plane $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e};\mathbf{z}) = 0$ gives the 2D EXIT chart for the turbo decoder alone. This 2D EXIT chart in Fig. 4(a) shows that without the aid of the UEC trellis decoder, the decoding trajectory would converge at about $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}) = 0.08$. By contrast, when the $r = 6$-state UEC trellis decoder is introduced, an open 3D tunnel can be created, as shown in Fig. 5(b). This allows the iterative decoding trajectory to reach the extrinsic MIs of $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\rm l}^{\rm e};\mathbf{z}) = 1$, where low decoding error rates are facilitated, as described in Section III-A1. Note that the trajectories shown in Fig. 5 rely on the fixed, periodic decoder activation order of {URC1, URC2, UEC; URC1, URC2, UEC; ...}, as described in Section II-B. However, the 3D EXIT chart in Section III-C will be used for comparing the quantitative potential benefits associated with activating each decoder at each stage of the iterative decoding process, in order to dynamically adapt the decoder activation order for expediting the attainable convergence.

As shown in Fig. 5(c), a wider EXIT chart tunnel is created at $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.514$, requiring fewer activations of the UEC, URC1 and URC2 decoders in order to achieve a low decoding error rate. By contrast, the EXIT chart tunnel is closed for $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.378$, preventing the achievement of a low decoder error rate, as shown in Fig. 5(a). In Section III-A3, our 3D EXIT charts are projected into two dimensions, in order to characterize the effect of the $I(\tilde{\mathbf{w}};\mathbf{w})$ value upon the iterative decoding convergence of the proposed UEC-Turbo scheme, as well as to evaluate its capacity-achieving capability.

3) 2D EXIT chart projections: The 3D EXIT surfaces of Fig. 5 can be projected into 2D [10], [11], [16], which were plotted in the 2D EXIT charts of Fig. 4. More explicitly,

Fig. 4(a) complements the URC1 component's EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{u}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ with the EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}) = f_{\mathrm{o,l}}[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w}), r]$ - where $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.405$ - which represents a 2D projection of the 3D UEC and URC2 EXIT surfaces of Fig. 5. This EXIT function characterizes the process in which the LLR vector $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}$ is fed into the UEC and URC2 decoders, where these components are iteratively operated until convergence is achieved, and then they generate the LLR vector $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}}$, as shown in Fig. 1. As a result, Fig. 4(a) characterizes a decoder activation order of {URC1, URC2, UEC, URC2, UEC, ..., URC2, UEC; URC1, URC2, UEC, URC2, UEC, ..., URC2, UEC; ...}. Note that the 2D projection of the 3D UEC and URC1 EXIT surfaces is identical to that of the 3D UEC and URC2 EXIT surfaces, owing to the symmetry of URC1 and URC2. Fig. 4(a) shows that an open EXIT chart tunnel is created at $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.405$, when $r = 6$ states are employed in the UEC trellis decoder, which is in agreement with the 3D EXIT chart of Fig. 5(b). This facilitates iterative decoding convergence to the $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{a}};\mathbf{z}) = I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}};\mathbf{z}) = 1$ point at the top-right corner of the EXIT chart of Fig. 4(a), where a low decoding error rate is achieved.

Similarly, the inverted UEC EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}), r]$ of Fig. 4(b) can be complemented with the EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = f_{\mathrm{u,l}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$, which represents a 2D projection of the 3D URC1 and URC2 EXIT surfaces of Fig. 5. This EXIT function characterizes the process, in which the LLR vector $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}$ is forwarded to the URC1 and URC2 components of Fig. 1, where these components are iteratively operated until convergence is achieved, and then they generate the LLR vector $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}}$, as shown in Fig. 1. As a result, Fig. 4(b) characterizes a component activation order of {URC1, URC2, URC1, URC2, ..., URC1, URC2, UEC; URC1, URC2, URC1, URC2, ..., URC1, URC2, UEC; ...}. Note that in agreement with the 3D EXIT chart of Fig. 5(b), an open EXIT chart tunnel is created in Fig. 4(b) at $I(\tilde{\mathbf{w}};\mathbf{w}) = 0.405$, when $r = 6$ states are employed in the UEC trellis decoder. This open tunnel facilitates iterative decoding convergence to the top edge of the EXIT chart, where we have $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = 1$.

The area properties of 2D EXIT charts[1] [24] can be employed to determine whether an iteratively decoded scheme suffers from any capacity loss, which prevents its near-capacity operation. As discussed in Section II-A and shown in Table II, the effective throughput of the proposed UEC-Turbo scheme is $\eta = 0.762$ information bits/symbol. When communicating over an uncorrelated narrowband Rayleigh fading channel, the Discrete-input Continuous-output Memoryless Channel (DCMC) capacity C becomes equal to η, when the E_{b}/N_0 value is equal to the *capacity bound* of 0.84 dB, as shown in Table II. This implies that reliable communication is possible for E_{b}/N_0 values above 0.84 dB, provided that an open tunnel can be created in the EXIT chart of Fig. 4(b).

However, in order to facilitate an open EXIT chart tunnel, it is necessary, but not sufficient, for the area A_{o} beneath the inverted UEC EXIT function $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}) = f_{\mathrm{o}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}), r]$ to exceed the area A_{i} beneath the 2D projection of the URC1 and URC2 EXIT surfaces $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{a}};\mathbf{z}) = f_{\mathrm{u,l}}[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}};\mathbf{z}), I(\tilde{\mathbf{w}};\mathbf{w})]$ [24]. For the cases, where the proposed UEC-Turbo scheme of Fig. 1 employs $r \in \{2, 4, 6, 8\}$ UEC trellis states, Table II quantifies the *area bound* as the E_{b}/N_0 value at which $A_{\mathrm{o}} = A_{\mathrm{i}}$. These area bounds represent the lowest E_{b}/N_0 values, where it would be possible to achieve a low decoding error rate, provided that the EXIT functions matched each other sufficiently well.

Note that the area bound exceeds the capacity bound for all cases considered in Table II, where the discrepancy represents *capacity loss*. However, this capacity loss can be seen to approach zero as the number of UEC trellis states r is increased. This is because the expression for A_{o} of [6, Equation (14)] asymptotically approaches the UEC coding rate R_{o} as the number of trellis states r is increased [6], where $A_{\mathrm{o}} = R_{\mathrm{o}}$ is a necessary condition for avoiding capacity loss [25]. Furthermore, the other necessary condition $A_{\mathrm{i}} = C/[R_{\mathrm{i}} \cdot \log_2(M)]$ is also satisfied, since the URC1 and URC2 codes of Fig. 1 each have an individual coding rate of $2R_{\mathrm{i}} = 1$ [25].

In practice, the EXIT functions of Fig. 4(b) do not match each other perfectly and hence no an open tunnel emerges, until the E_{b}/N_0 value exceeds the *tunnel bound*. As shown in Fig. 4(b) and confirmed in Fig. 4(a) and Fig. 5(b), the proposed UEC-Turbo scheme has a tunnel bound of $E_{\mathrm{b}}/N_0 = 1.3$ dB, when employing $r = 6$ UEC trellis states. The tunnel bounds for $r \in \{2, 4, 6, 8\}$ are summarised in Table II.

B. Decoding complexity

In this section, we quantify the complexity cost of activating the UEC trellis decoder when employing r number of trellis states, in addition to quantifying the complexity cost associated with activating the URC1 and URC2 decoders. In [26], the complexity of each component decoder was quantified in the context of an iterative decoding scheme using the number of transitions and states in their respective trellises. This was justified, since all components in the considered schemes were of the same type. However, in our UEC-Turbo scheme, the complexity of the UEC decoder may differ from that of the URC decoders, even if they have the same number of transitions or states in their trellises. Therefore, our approach is to decompose each of the BCJR calculations into their constituent addition and max* operations, allowing a fair comparison between the complexities of components having different types.

Owing to this, our approach is to decompose each γ_t, α_t, β_t and extrinsic LLR calculation into their constituent addition and max* operations. This allows a fair comparison between the complexities of components having the above described complications. More specifically, we quantify these costs in terms of the computational complexity by observing that each of these decoders operates on the basis of only addition and $\max^*$ operations, where we have [27]

$$\max^*(\tilde{z}_1, \tilde{z}_2) = \max(\tilde{z}_1, \tilde{z}_2) + \ln(1 + e^{-|\tilde{z}_1 - \tilde{z}_2|}). \quad (9)$$

The complexity of each type of decoder scales linearly with the number of bits bn in the encoded vector $\mathbf{z}$ of

[1]It was shown by Land [23] that the size of the open area between a pair of components exchanging extrinsic information is commenu with the associated capacity loss.

Fig. 1. Therefore, Table III lists the number of $\max^*$ and addition operations that are performed per bit of $\mathbf{z}$, when each type of decoder employs a trellis having r states. Here, the computational complexity of the UEC trellis decoder depends on whether it is used for generating the LLR vector $\tilde{\mathbf{z}}^{\mathrm{e}}$ or $\tilde{\mathbf{y}}^{\mathrm{p}}$.

TABLE III: Number of $\max^*$ and addition operations that are performed per bit of $\mathbf{z}$, for various types of decoder employing trellises having r states.

Decoder	r	$\max^*$	add	ACS
$n=1$-bit Trellis BCJR decoder $\tilde{\mathbf{y}}^{\mathrm{p}}$	2	6	19	49
	4	14	37	107
	6	22	55	165
	8	30	73	223
	10	38	91	281
$n=1$-bit Trellis BCJR decoder $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}$	2	6	20	50
	4	14	38	108
	6	22	56	166
	8	30	74	224
	10	38	92	282
URC BCJR decoder	2	6	19	49
	8	30	73	223
URC Viterbi decoder	2	2	8	18

In practice, the second term $f_{\mathrm{c}} = \ln(1 + e^{-|\tilde{z}_1 - \tilde{z}_2|})$ in the $\max^*$ operation of (9) can be implemented at a low computational complexity by employing a Look-Up-Table (LUT) [27]. When employing an 8-entry LUT, as few as $\log_2(8) = 3$ compare operations are required for selecting the particular LUT entry that best approximates f_{c}. Furthermore, a compare operation is required for computing $\max(\tilde{z}_1, \tilde{z}_2)$ and an addition operation is required for evaluating $\max(\tilde{z}_1, \tilde{z}_2) + f_{\mathrm{c}}$. Therefore, each $\max^*$ operation can be considered to be equivalent to five ACS operations. By contrast, each addition operation corresponds to a single ACS operation. Using this logic, Table III lists the total number of ACS operations that are performed by each type of decoder.

C. Dynamic adjustment of the decoder activation order

In this section, we propose a novel adaptive iterative decoding technique for dynamically adjusting the decoder activation order in the UEC-Turbo scheme of Fig. 1. In contrast to the fixed decoder activation orders that have been discussed in Sections II-B, III-A2 and III-A3, these dynamic decoder activation orders can vary between frames. More specifically, following the activation of a URC decoder, our novel adaptive iterative decoding technique can either activate the other URC decoder, or it can activate the UEC trellis decoder and select the number $r \in \{2, 4, 6, 8\}$ of trellis states to employ. Similarly, following the activation of the UEC trellis decoder, our novel adaptive iterative decoding technique can then activate either the URC1 or the URC2 decoder. As we will show in Section IV, this expedites iterative decoding convergence, facilitating near-capacity operation, with reduced computational complexity. We employ the EXIT charts of Section III-A to quantify the benefit associated with activating the UEC, URC1 and URC2 decoders at each stage of the iterative decoding process, as well as the computational complexity of Section III-B to quantify the corresponding cost.

At each stage of the iterative decoding process, our proposed technique estimates the quality of the LLR vectors $\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}$, $\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}$ and $\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}$ that were obtained after the most recent activation of the respective decoders. More specifically, the averaging method of the measuring MI [28] may be employed for estimating $I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}; \mathbf{z})$, $I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}; \mathbf{z})$ and $I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}; \mathbf{z})$, without requiring the explicit knowledge of the bit values in the vector $\mathbf{z}$. Likewise, the averaging method may be employed for estimating the MI $I(\tilde{\mathbf{w}}; \mathbf{w})$ of the LLRs $\tilde{\mathbf{w}}$ provided by the QPSK demodulator. Following this, the 3D EXIT charts of Section III-A2 may be employed to predict the extrinsic MI that is offered by activating each decoder. Then, the corresponding MI improvements can be obtained as the discrepancies between these predicted MIs and the current MIs, in order to quantify the benefit associated with activating each decoder. For example, the benefit associated with activating the UEC trellis decoder using r states may be quantified by the MI improvement of

$$\Delta I_{\mathrm{o}}^{r} = f_{\mathrm{o}}\left[I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}; \mathbf{z}), r\right] - I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}; \mathbf{z}). \tag{10}$$

Similarly, the benefits associated with the URC1 and URC2 decoders may be quantified as

$$\Delta I_{\mathrm{u}} = f_{\mathrm{u}}\left[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}; \mathbf{z}), \tag{11}$$

$$\Delta I_{\mathrm{l}} = f_{\mathrm{l}}\left[I(\tilde{\mathbf{z}}_{\mathrm{o}}^{\mathrm{e}}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\mathrm{u}}^{\mathrm{e}}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\mathrm{l}}^{\mathrm{e}}; \mathbf{z}), \tag{12}$$

respectively.

As discussed in Section III-A3, when the number of states r employed by the UEC trellis decoder is increased, a MI improvement $\Delta I_{\mathrm{o}}^{r}$ may be expected. However, this will be achieved at the cost of an increased computational complexity. Therefore, our novel adaptive iterative decoding technique also considers the computational complexity cost C_{o}^{r}, C_{u} and C_{l} of the UEC decoder having $r \in \{2, 4, 6, 8\}$ states, as well as the 8-state URC1 and URC2 decoders, respectively. As discussed in Section III-B, Table III states the number of ACS operations required by each decoder. Note that while our adaptive technique makes *on-line* decisions about which decoder to activate at each stage of the iterative decoding process, this is performed using the complexity figures of Table III, which were calculated in an *off-line* fashion. These complexity figures remain constant throughout each decoding iteration and for each transmitted frame. For example, when employing the $r = 6$-state $n = 1$-bit trellis shown in Fig. 3, the complexity cost is $\mathrm{C}_{\mathrm{o}}^{6} = 166$ ACS operations per bit of $\mathbf{z}$.

After each stage of the iterative decoding process, the benefit-to-cost ratios $\Delta I_{\mathrm{o}}^{r}/C_{\mathrm{o}}^{r}$, $\Delta I_{\mathrm{u}}/C_{\mathrm{u}}$ and $\Delta I_{\mathrm{l}}/C_{\mathrm{l}}$ are calculated for the UEC trellis decoder having $r \in \{2, 4, 6, 8\}$ states, as well as for the URC1 and URC2 decoders, respectively. Then, the particular decoder offering the highest benefit-to-cost ratio is selected for activation in the next stage of the iterative decoding process.

As shown in Algorithm 1, the adaptive iterative decoding process continues, until the accumulated complexity C of the activated components reaches the pre-defined complexity limit $C_{\mathrm{limit}} = 273$ ACS operations. Note that these 273 ACS operations are reserved, so that even if the component having the highest complexity is selected in the final stage of the iterative decoding process, there will be 49 ACS operations

Algorithm 1 Adaptive iterative decoding algorithm

$\tilde{\mathbf{z}}_{\rm o}^{\rm e} \leftarrow \mathbf{0}$, $\tilde{\mathbf{z}}_{\rm u}^{\rm e} \leftarrow \mathbf{0}$, $\tilde{\mathbf{z}}_{\rm l}^{\rm e} \leftarrow \mathbf{0}$, $C \leftarrow 0$.
while $C \leq C_{\rm limit} - 273$ **and** $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm u}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z}) < 0.999$ **do**
  $\tilde{\mathbf{z}}_{\rm o}^{\rm a} \leftarrow \tilde{\mathbf{z}}_{\rm u}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}$, $\tilde{\mathbf{z}}_{\rm u}^{\rm a} \leftarrow \tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}$ and $\tilde{\mathbf{z}}_{\rm l}^{\rm a} \leftarrow \tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm u}^{\rm e}$.
  Measure MI: $I(\tilde{\mathbf{z}}_{\rm o}^{\rm a}; \mathbf{z})$, $I(\tilde{\mathbf{z}}_{\rm u}^{\rm a}; \mathbf{z})$ and $I(\tilde{\mathbf{z}}_{\rm l}^{\rm a}; \mathbf{z})$.
  for $r \in \{2, 4, 6, 8\}$ **do**
    $\Delta I_{\rm o}^r \leftarrow f_{\rm o}\left[I(\tilde{\mathbf{z}}_{\rm o}^{\rm a}; \mathbf{z}), r\right] - I(\tilde{\mathbf{z}}_{\rm o}^{\rm e}; \mathbf{z})$. ▷ *Predict MI improvement*
  end for
  $\Delta I_{\rm u} \leftarrow f_{\rm u}\left[I(\tilde{\mathbf{z}}_{\rm u}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\rm u}^{\rm e}; \mathbf{z})$.
  $\Delta I_{\rm l} \leftarrow f_{\rm l}\left[I(\tilde{\mathbf{z}}_{\rm l}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z})$.
  $best \leftarrow max(\Delta I_{\rm o}^r / C_{\rm o}^r, \Delta I_{\rm u}/C_{\rm u}, \Delta I_{\rm l}/C_{\rm l})$, $r \in \{2, 4, 6, 8\}$.
  for $r \in \{2, 4, 6, 8\}$ **do**
    if $\Delta I_{\rm o}^r / C_{\rm o}^r = best$ **then**
      Activate r-state UEC trellis decoder to convert $\tilde{\mathbf{z}}_{\rm o}^{\rm a}$ into $\tilde{\mathbf{z}}_{\rm o}^{\rm e}$,
      $C \leftarrow C + C_{\rm o}^r$.
    end if
  end for
  if $\Delta I_{\rm u}/C_{\rm u} = best$ **then**
    Activate URC1 decoder to convert $\tilde{\mathbf{z}}_{\rm u}^{\rm a}$ into $\tilde{\mathbf{z}}_{\rm u}^{\rm e}$,
    $C \leftarrow C + C_{\rm u}$.
  end if
  if $\Delta I_{\rm l}/C_{\rm l} = best$ **then**
    Activate URC2 decoder to convert $\tilde{\mathbf{z}}_{\rm l}^{\rm a}$ into $\tilde{\mathbf{z}}_{\rm l}^{\rm e}$,
    $C \leftarrow C + C_{\rm l}$.
  end if
end while
$\tilde{\mathbf{z}}_{\rm o}^{\rm a} \leftarrow \tilde{\mathbf{z}}_{\rm u}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}$
if $C \leq C_{\rm limit} - 223$ **then**
  Activate $r = 8$-state UEC trellis decoder to convert $\tilde{\mathbf{z}}_{\rm o}^{\rm a}$ into $\tilde{\mathbf{y}}^{\rm p}$,
else if $C \leq C_{\rm limit} - 165$ **then**
  Activate $r = 6$-state UEC trellis decoder to convert $\tilde{\mathbf{z}}_{\rm o}^{\rm a}$ into $\tilde{\mathbf{y}}^{\rm p}$,
else if $C \leq C_{\rm limit} - 107$ **then**
  Activate $r = 4$-state UEC trellis decoder to convert $\tilde{\mathbf{z}}_{\rm o}^{\rm a}$ into $\tilde{\mathbf{y}}^{\rm p}$,
else
  Activate $r = 2$-state UEC trellis decoder to convert $\tilde{\mathbf{z}}_{\rm o}^{\rm a}$ into $\tilde{\mathbf{y}}^{\rm p}$.
end if

remaining. This then allows the $r = 2$ UEC trellis decoder to obtain the *a posteriori* LLR vector $\tilde{\mathbf{y}}^{\rm p}$, as shown in Table III. However, if more ACS operations are available following the final stage of iterative decoding, then a UEC trellis having a higher number of states r can be selected for obtaining $\tilde{\mathbf{y}}^{\rm p}$ at a reduced probability of error. Furthermore, the iterative decoding process will be terminated early, if the *a posteriori* MI $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e} + \tilde{\mathbf{z}}_{\rm u}^{\rm e} + \tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z})$ reaches 0.999, which implies that error free decoding can be achieved without operating any further component decoders.

D. Storage analysis

In this section, we consider the storage requirements of the novel adaptive iterative decoding technique introduced in Section III-C. In order to quantify the benefits associated with activating each decoder, in comparison to that of the others, the value of the functions $f_{\rm o}\left[I(\tilde{\mathbf{z}}_{\rm u}^{\rm e}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z}), r\right]$, $f_{\rm u}\left[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right]$ and $f_{\rm l}\left[I(\tilde{\mathbf{z}}_{\rm o}^{\rm e}; \mathbf{z}), I(\tilde{\mathbf{z}}_{\rm u}^{\rm e}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right]$ is required. Rather than storing these 3D EXIT functions, the associated storage requirements can be reduced by storing the 2D EXIT functions $f_{\rm o}\left[I(\tilde{\mathbf{z}}_{\rm o}^{\rm a}; \mathbf{z}), r\right]$, $f_{\rm u}\left[I(\tilde{\mathbf{z}}_{\rm u}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right]$ and $f_{\rm l}\left[I(\tilde{\mathbf{z}}_{\rm l}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right]$. In this way, (10) – (12) may be reformulated as

$$\Delta I_{\rm o}^r = f_{\rm o}\left[I(\tilde{\mathbf{z}}_{\rm o}^{\rm a}; \mathbf{z}), r\right] - I(\tilde{\mathbf{z}}_{\rm o}^{\rm e}; \mathbf{z}), \quad (13)$$

$$\Delta I_{\rm u} = f_{\rm u}\left[I(\tilde{\mathbf{z}}_{\rm u}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\rm u}^{\rm e}; \mathbf{z}), \quad (14)$$

$$\Delta I_{\rm l} = f_{\rm l}\left[I(\tilde{\mathbf{z}}_{\rm l}^{\rm a}; \mathbf{z}), I(\tilde{\mathbf{w}}; \mathbf{w})\right] - I(\tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z}), \quad (15)$$

where $I(\tilde{\mathbf{z}}_{\rm o}^{\rm a}; \mathbf{z})$, $I(\tilde{\mathbf{z}}_{\rm u}^{\rm a}; \mathbf{z})$ and $I(\tilde{\mathbf{z}}_{\rm l}^{\rm a}; \mathbf{z})$ may be obtained using (8).

Furthermore, an infinite amount of storage would be required for storing the values of the 2D EXIT functions for all possible values of $I(\tilde{\mathbf{z}}_{\rm o}^{\rm e}; \mathbf{z})$, $I(\tilde{\mathbf{z}}_{\rm u}^{\rm e}; \mathbf{z})$, $I(\tilde{\mathbf{z}}_{\rm l}^{\rm e}; \mathbf{z})$ and $I(\tilde{\mathbf{w}}; \mathbf{w})$. Instead, we recommend storing the 2D EXIT functions for only a limited set of carefully selected quantised values of these parameters. During the adaptive iterative decoding process, the values of the 2D EXIT functions may be approximated by interpolating between the values stored.

In particular, we recommend storing the values of the 2D UEC EXIT function for only the 4 values of r in the set $\{2, 4, 6, 8\}$, since having $r \geq 10$ states offer only a marginal additional MI improvement compared to $r = 8$, at the cost of a significantly higher computational complexity. Additionally, it is only necessary to store one set of 2D URC EXIT functions, since the URC1 and URC2 decoders are identical. We recommend storing the values of the 2D URC EXIT function for only the 24 $I(\tilde{\mathbf{w}}; \mathbf{w})$ values in the set of $\{0.337$, 0.362, 0.389, 0.415, 0.443, 0.456, 0.465 0.476, 0.487, 0.498, 0.510, 0.521, 0.532, 0.543, 0.554, 0.565, 0.576, 0.587, 0.598, 0.609, 0.635, 0.661, 0.686, $0.710\}$, which correspond to the $E_{\rm b}/N_0$ values of $\{0.0, 0.5, 1.0, 1.5\}$, $\{2.0, 2.2, \ldots, 4.8, 5.0\}$ and $\{5.5, 6.0, 6.5, 7.0\}$. This is motivated by the observation that at very low $I(\tilde{\mathbf{w}}; \mathbf{w})$ values, iterative decoding convergence is impossible and hence any efforts to optimise the process are futile. Furthermore, at very high $I(\tilde{\mathbf{w}}; \mathbf{w})$ values, iterative decoding is unnecessary, since error free decoding can typically be achieved by activating each of the UEC, URC1 and URC2 decoders only once. Similarly, we recommend storing the values of the 2D EXIT functions for only the 26 values of *a priori* MI in the set $\{0.00, 0.04, \ldots, 0.96, 1.00\}$. Finally, we suggest using 8 bits to store the extrinsic MI values that are obtained by the 2D EXIT functions. In Section IV, we will show that using more storage would not give significantly improved performance, while reducing it would degrade the performance.

Following the above recommendations, a total of 5824 bits memory is required for storing the 2D EXIT functions. We consider this amount of storage to be practical, since it is comparable to the number of bits required to store the set of 3GPP LTE interleaver parameters [29].

IV. Comparison with Benchmarkers

In this section, we compare our Adaptive UEC-Turbo scheme of Section III with four benchmarkers, which are listed in Table II.

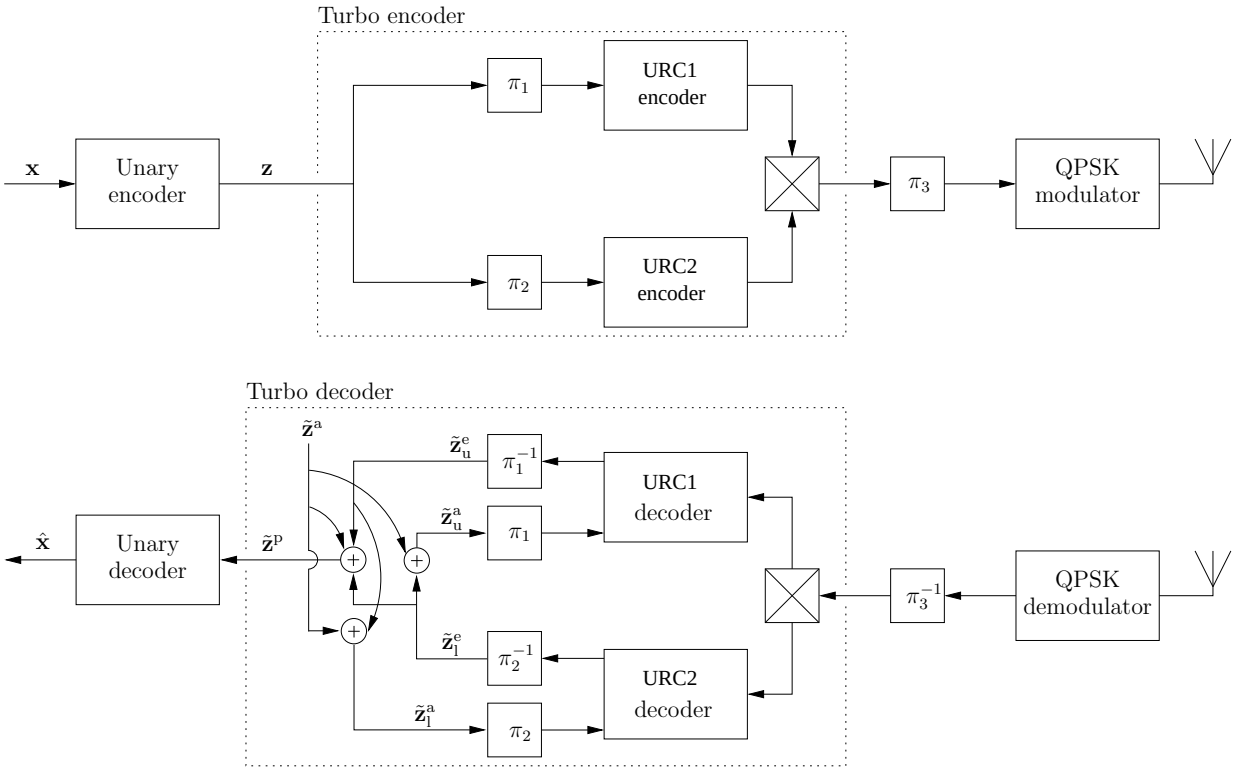

Fig. 6: Schematic of the Unary-Turbo scheme, in which unary coding is serially concatenated with turbo coding and Gray-coded QPSK modulation schemes. Here, π_1, π_2 and π_3 represent interleavers, while π_1^{-1}, π_2^{-1} and π_3^{-1} represent the corresponding deinterleavers. Multiplexer and de-multiplexer operations are also employed before π_3 and after π_3^{-1}, respectively.

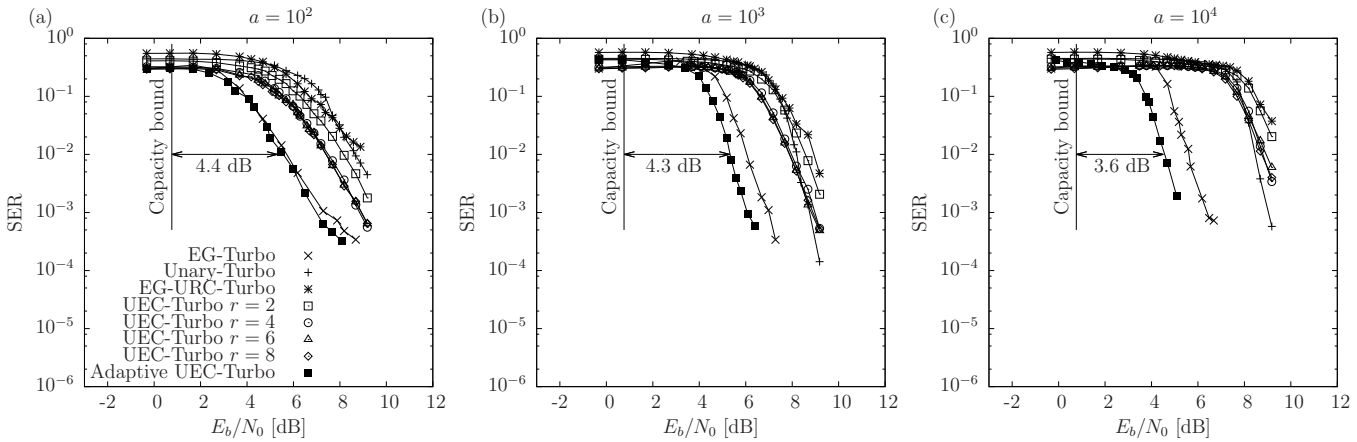

Fig. 7: The SER performance of the five schemes of Table II, when conveying vectors comprising (a) $a = 10^2$, (b) $a = 10^3$ and (c) $a = 10^4$ symbols, which obey a zeta distribution having the parameter $p_1 = 0.797$. Communication is over an uncorrelated narrowband Rayleigh fading channel and a complexity limit of $C_{\mathrm{limit}} = 1948$ ACS operations is imposed for decoding each of the bn bits in the vector $\mathbf{z}$. Different random designs are employed for the interleavers π_1, π_2 and π_3 in each simulated frame.

1) Non-adaptive UEC-Turbo
In the first benchmarker, we consider the non-adaptive UEC-Turbo scheme, employing the fixed, periodic decoder activation order of {URC1, URC2, UEC; URC1, URC2, UEC; ...}. In both the transmitter and receiver of this non-adaptive scheme, we employ an $n = 1$-bit UEC trellis having a fixed number of states $r = 2$, 4, 6 or 8, corresponding to the UEC codebooks $\mathbb{C} = \{1\}$, $\mathbb{C} = \{1, 1\}$, $\mathbb{C} = \{1, 1, 1\}$ and $\mathbb{C} = \{1, 1, 1, 1\}$, respectively.
2) EG-URC-Turbo
Additionally, the JSCC UEC-Turbo scheme of Fig. 1 may be compared to a classic SSCC benchmarker, which we refer to as the EG-URC-Turbo scheme. Separately from channel coding, this scheme employs an EG code for source coding, where the first ten EG codewords are illustrated in Table I and the average EG codeword length is given by $l = \sum_{x \in \mathbb{N}_1} P(x) \left(2\lfloor \log_2(x) \rfloor + 1\right)$.
In analogy to [6, Fig. 2(b)], the EG-URC-Turbo transmitter schematic may be obtained by replacing the unary encoder of Fig. 1 with an EG encoder. Furthermore, the trellis encoder of Fig. 1 is replaced by an accumulator, which we refer to as the $r = 2$-state $n = 1$-bit URC3 encoder. This accumulator is included in order to convert the vector of non-equiprobable bits $\mathbf{y}$ into the vector of equiprobable bits $\mathbf{z}$.
More explicitly, the b-bit vector $\mathbf{y} = [y]_{j=1}^{b}$ may be modeled as a realization of a vector $\mathbf{Y} = [Y_j]_{j=1}^{b}$ comprising b binary RVs, where $\Pr(Y_j = 0) \neq \Pr(Y_j = 1)$ and the bit entropy is $H_{Y_j} = H[\Pr(Y_j = 0)] + H[\Pr(Y_j = 1)] < 1$ in general. Owing to the recursive infinite impulse response nature of the accumulator, the encoded bits $\mathbf{z}$ have a bit entropy of $H_{Z_k} = 1$, which is necessary for avoiding capacity loss, as we will show below. In the receiver, the trellis decoder of Fig. 1 is replaced by the URC3 decoder.
During iterative decoding, the fixed decoder activation order {URC1, URC2, URC3; URC1, URC2, URC3; ...} is employed, which is justified since the $r = 2$-state URC3 decoder has a significantly lower decoding complexity than the $r = 8$-state URC1 and URC2 decoders. As in Fig. 1, the URC3 decoder is employed to convert the *a priori* LLR vector $\tilde{\mathbf{z}}^{\mathrm{a}}$ into the extrinsic LLR vector $\tilde{\mathbf{z}}^{\mathrm{e}}$ using the BCJR decoding algorithm. However, in analogy to [6, Fig. 2(b)], this process is assisted by additionally considering the *a priori* LLR vector $\tilde{\mathbf{y}}^{\mathrm{a}} = [\tilde{y}_j^{\mathrm{a}}]_{j=1}^{b}$, where $\tilde{y}_j^{\mathrm{a}} = \ln[\Pr(Y_j = 0)/\Pr(Y_j = 1)]$. In the final decoding iteration, the URC decoder uses the Viterbi algorithm [30] for generating the bit sequence $\hat{\mathbf{y}}$, which may be subsequently decoded by the EG decoder.
3) Unary-Turbo
In order to illustrate that satisfying the $H_{Z_k} = 1$ constraint is necessary for avoiding capacity loss, we consider additional SSCC benchmarkers, in which the bits input to the turbo encoder are not equiprobable, giving $H_{Z_k} < 1$.
In the Unary-Turbo benchmarker of Fig. 6, a unary source encoder is concatenated with a separate turbo channel encoder. This benchmarker may be considered to be an adaptation of the UEC-Turbo scheme of Fig. 1, in which the UEC trellis encoder is omitted. Owing to this, we have $\mathbf{z} = \mathbf{y}$, $\Pr(Z_k = 0) = \Pr(Y_j = 0)$, $\Pr(Z_k = 1) = \Pr(Y_j = 1)$ and $H_{Z_k} = H_{Y_j}$, where the bit indices k and j are interchangeable, since we have $n = 1$ bit in $\mathbf{z}$ per bit in $\mathbf{y}$.
In the receiver of Fig. 6, the iterative decoding process exploits some of the residual redundancy present within the bit vector $\mathbf{z}$ by adding the *a priori* LLRs vector $\tilde{\mathbf{z}}^{\mathrm{a}} = [\tilde{z}_k^{\mathrm{a}}]_{k=1}^{bn}$ to the extrinsic LLR vectors, where we have $\tilde{z}_k^{\mathrm{a}} = \ln[\Pr(Z_k = 0)/\Pr(Z_k = 1)]$. Following the final decoding iteration, the *a posteriori* LLR vector $\tilde{\mathbf{z}}^{\mathrm{p}}$ of Fig. 6 can be soft-decision unary decoded, as described in Section II-B.
4) EG-Turbo
Similarly, our final SSCC benchmarker can be obtained by replacing the unary code of Fig. 6 with an EG code. The resultant EG-Turbo scheme represents a specific version of the EG-URC-Turbo benchmarker, in which the bits forwarded to the turbo encoder are not equiprobable, giving $H_{Z_k} < 1$. Note that following the final decoding iteration in the EG-Turbo receiver, it is necessary to use hard decisions to convert the *a posteriori* LLR vector $\tilde{\mathbf{z}}^{\mathrm{p}}$ into the bit vector $\hat{\mathbf{z}}$, before performing EG decoding.

For all five schemes, symbol values $\mathbf{x}$ that obey a zeta distribution having a parameter value of $p_1 = 0.797$ are employed. This value offers a fair comparison, since [6, Fig. 5] shows that unary and EG codes provide an equal average codeword length l for $p_1 = 0.797$. As a result, all schemes have the same effective throughput of $\eta = 0.762$ information bits/symbol, when $n = 1$-bit codewords and $M = 4$-ary QPSK are employed, as shown in Table II. Note that while all considered schemes have the same effective throughput η, they have different outer and inner coding rates R_{o} and R_{i}, according to (6) and (7), respectively. This is because the various schemes considered have different bit entropies H_{Z_k}.

Table II compares the DCMC capacity, area and tunnel bounds of the various considered schemes, when employing QPSK modulation for communication over an uncorrelated narrowband Rayleigh fading channel. Note that all considered schemes have the same capacity bound of 0.84 dB, since they all have the same effective throughput of 0.762 information bits/symbol. The area bounds of the UEC-Turbo scheme were determined as detailed in Section III-A3. This technique was also employed for determining the area bound of the EG-URC-Turbo schemes, although the second equation of [6, Section VI-B] was employed for determining the area beneath the inverted EG-CC EXIT function, rather than [6, Equation (14)]. The area bounds of the Unary-Turbo and EG-Turbo schemes were obtained by plotting the corresponding EXIT charts, in which the measured MI can assume values in the range $[0, H_{Z_k}]$. Similarly, plots of the corresponding EXIT charts were employed for determining the tunnel bounds in the case of all the schemes considered.

As described in Section III-A3, the capacity loss imposed by the proposed UEC-Turbo scheme becomes vanishingly small as the number of UEC trellis states r is increased. Owing to this benefit of JSCC, the proposed UEC-Turbo scheme employing $r \geq 6$ UEC trellis states suffers from less capacity loss than the SSCC EG-URC-Turbo benchmarker, as shown in Table II. Furthermore, the proposed UEC-Turbo scheme can be seen to impose a lower capacity loss than the SSCC Unary-Turbo benchmarker. As described in Section III-A3, the proposed UEC-Turbo scheme accrues this benefit by conditioning the bits in the vector $\mathbf{z}$ so that they adopt equiprobable values, giving $H_{Z_k} = 1$ and $2R_\mathrm{i} = 1$. By contrast, the bits in the vector $\mathbf{z}$ do not adopt equiprobable values in the Unary-Turbo benchmarker, giving $H_{Z_k} < 1$. As described in Section III-A3, this corresponds to URC1 and URC2 individual coding rates of $2R_\mathrm{i} < 1$, which imposes a capacity loss [25]. This phenomenon also explains why the $H_{Z_k} = 1$ EG-URC-Turbo benchmarker imposes a 0.91 dB lower capacity loss than the $H_{Z_k} < 1$ EG-Turbo benchmarker, as shown in Table II.

The SER of the schemes considered is compared in Fig. 7 for the case where vectors comprising $a = 10^2, 10^3, 10^4$ $p_1 = 0.797$ zeta-distributed symbols are transmitted over an uncorrelated narrowband Rayleigh fading channel. In order to facilitate fair comparisons amongst the schemes having different computational complexities, their iterative decoding was always terminated, when the complexity limit of 1948 ACS operations for each of bn bits in the vector $\mathbf{z}$ was reached by each scheme, as show in Algorithm 1. Fig. 7 shows that our proposed Adaptive UEC-Turbo scheme outperforms each of the four benchmarkers, regardless of whether $a = 10^2$, 10^3 or 10^4 symbols are conveyed by each frame. In conclusion, our proposed Adaptive UEC-Turbo scheme offers an SER of 10^{-2} at an E_b/N_0 value, which is nearly 1.2 dB lower than that required by the best benchmarker, when $a = 10^4$. This scheme is capable of operating within 3.6 dB of the DCMC capacity bound, as shown in Fig. 7.

Fig. 8 characterises the average number of times that each of the component decoders is activated, when the proposed adaptive UEC-Turbo scheme is employed to decode vectors comprising $a = 100$ symbols. This shows that at low E_b/N_0 values, the proposed adaptive iterative decoder relies on the URC decoders to a greater extent, but that as the E_b/N_0 value increases, the UEC decoder is selected more frequently. Note that similar observations were found when employing vectors comprising $a = 10^3$ and $a = 10^4$ symbols.

In Fig. 9, we show how the storage requirements of the proposed adaptive UEC-Turbo scheme affect the attainable SER performance. Here, we repeat the SER performance provided in Fig. 7(b) for vectors comprising $a = 10^3$ symbols, when employing the adaptive UEC-Turbo scheme having the 5824-bit storage requirement discussed in Section III-D. This scheme relies on the storage parameters of 8-bit quantization, 26 values of *a priori* MI and 24 values of $I(\tilde{\mathbf{w}}; \mathbf{w})$. Fig. 9 compares this SER performance to those of three schemes having lower storage requirements, namely schemes to C1, C2 and C3. Additionally, the attainable SER performance is compared to those of three schemes having higher storage

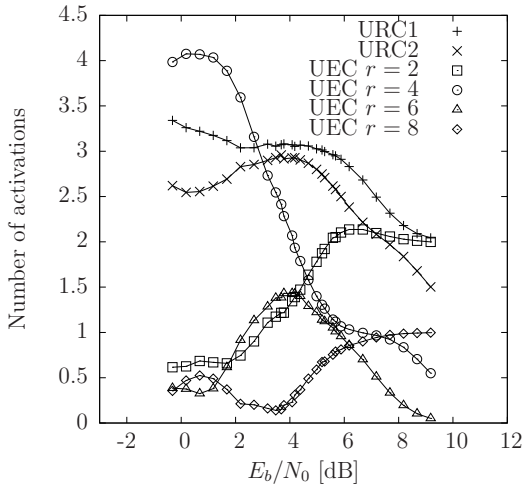

Fig. 8: Average number of activations of the component decoders in the proposed adaptive UEC-Turbo scheme, when conveying vectors comprising $a = 10^2$ symbols, which obey a zeta distribution having the parameter $p_1 = 0.797$. Communication is over an uncorrelated Rayleigh fading channel and a complexity limit of $C_\mathrm{limit} = 1948$ ACS operations is imposed for decoding each of the bn bits in the vector $\mathbf{z}$. Different random designs are employed for the interleavers π_1, π_2 and π_3 in each simulated frame.

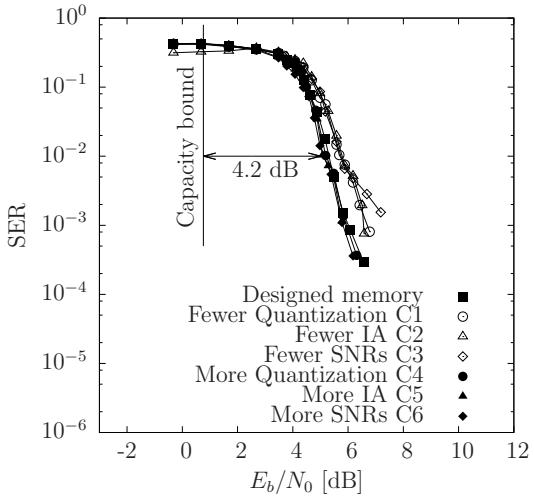

Fig. 9: The SER performance of the proposed adaptive UEC-Turbo scheme having various storage requirements, when conveying vectors comprising $a = 10^3$ symbols, which obey a zeta distribution having the parameter $p_1 = 0.797$. Communication is over an uncorrelated narrowband Rayleigh fading channel and a complexity limit of 1948 ACS operations is imposed for decoding each of the bn bits in the vector $\mathbf{z}$. Different random designs are employed for the interleavers π_1, π_2 and π_3 in each simulated frame.

requirements, namely schemes C4, C5 and C6. Each of these schemes employs a different setting for one of the three storage parameters, as follows:

C1: 5-bit quantization, giving a total storage requirement of 3640 bits.

C2: 11 values of *a priori* MI in the set $\{0.00, 0.10, \ldots, 0.90, 1.00\}$, having a total storage requirement of 2464 bits.

C3: 8 values of $I(\tilde{\mathbf{w}}; \mathbf{w})$ in the set {0.337, 0.389, 0.443, 0.498, 0.554, 0.609, 0.661, 0.710}, which correspond to the E_b/N_0 values of $\{0.0, 1.0, \ldots, 6.0, 7.0\}$, exhibiting a total storage requirement of 2496 bits.

C4: 9-bit quantization, giving a total storage requirement of 6552 bits.

C5: 51 values of *a priori* MI in the set $\{0.00, 0.02, \ldots, 0.98, 1.00\}$, imposing a total storage requirement of 11424 bits.

C6: 36 values of $I(\tilde{\mathbf{w}}; \mathbf{w})$ in the set {0.337, 0.347, 0.357, 0.367, 0.378, 0.389, 0.399, 0.410, 0.421, 0.432, 0.443, 0.456, 0.465 0.476, 0.487, 0.498, 0.510, 0.521, 0.532, 0.543, 0.554, 0.565, 0.576, 0.587, 0.598, 0.609, 0.620, 0.630, 0.640, 0.651, 0.661, 0.671, 0.681, 0.691, 0.700, 0.710}, which corresponds to the E_b/N_0 values of $\{0.0, 0.2, \ldots, 6.8, 7.0\}$, giving a total storage requirement of 8320 bits.

As shown in Fig. 9, the schemes having lower storage requirements than our recommendation in Section III-D Section III-C suffer from a degraded SER performance. By contrast, the schemes having higher storage requirements do not offer a significantly improved SER performance, motivating our recommendations. Note that similar observations were found, when employing vectors comprising $a = 10^2$ and $a = 10^4$ symbols.

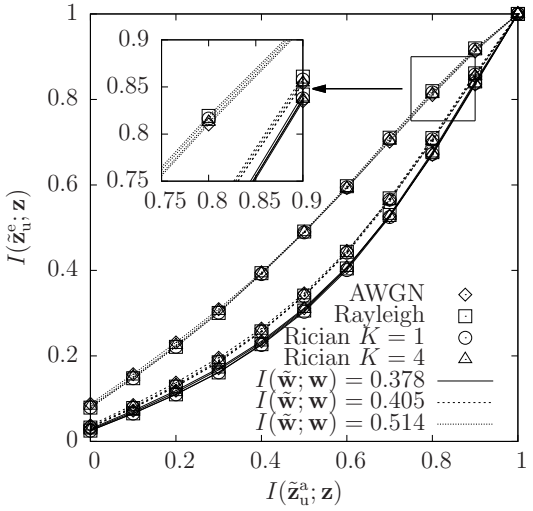

Fig. 10: 2D EXIT functions for the URC1 decoder in UEC-Turbo scheme, when transmitting over various uncorrelated narrowband channels, having the particular E_b/N_0 values that yield MI values of $I(\tilde{\mathbf{w}}; \mathbf{w}) = 0.378, 0.405$ and 0.514.

Furthermore, in order to generalise our adaptive decoding technique beyond the uncorrelated narrowband Rayleigh fading channel, we consider three additional channels models, namely the AWGN channel, as well as the Rician channel with

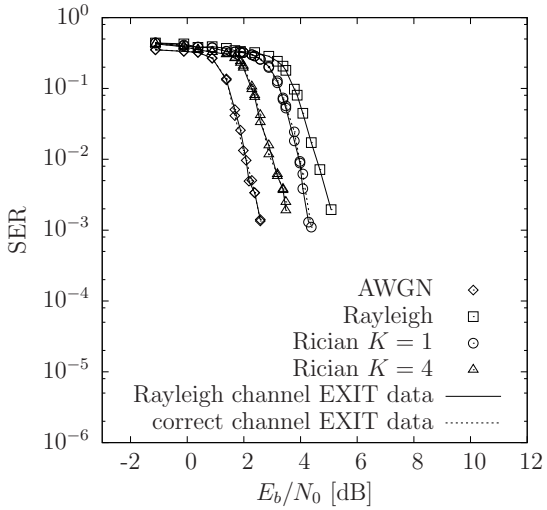

Fig. 11: The SER performance of the proposed adaptive UEC-Turbo scheme in various channels, when conveying vectors comprising $a = 10^4$ symbols, which obey a zeta distribution having the parameter $p_1 = 0.797$. Plots are provided for the case where the proposed adaptive iterative decoding technique employs 2D EXIT functions that were obtained using the correct channel model. Plots are also provided for the case of employing 2D EXIT functions that were obtained using the Rayleigh fading channel model.

K-factors of $K = 1$ and $K = 4$. Regardless of the channel model, Fig. 10 illustrates that the URC EXIT functions are quite similar to each other, when the MI $I(\tilde{\mathbf{w}}; \mathbf{w})$ happens to be the same. Owing to this, the proposed adaptive algorithm can be seamlessly employed for any of the above mentioned channels. In these cases, our technique can continue operating on the basis of the EXIT functions obtained using the Rayleigh fading model, with no performance penalty. This is demonstrated in Fig. 11, which compares the SER performance of our adaptive UEC-Turbo scheme, when employing EXIT functions obtained using the correct channel model and when employing those obtained using the Rayleigh fading model. As shown in Fig. 11, the SER performance is almost identical, regardless of whether the correct channel model is employed or not.

V. Conclusions

In this paper, we have demonstrated that the number of trellis states r employed by a UEC decoder can be dynamically adapted, in order to strike an attractive trade-off between its decoding complexity and its error correction capability. Furthermore, we have proposed the employment of 3D EXIT charts for controlling this dynamic adaptation, as well as for controlling the decoder activation order. This has been demonstrated for the scenario where the UEC code is serially concatenated with a turbo code, as shown in Fig. 1. In this way, we have expedited iterative decoding convergence, facilitating up to 1.2 dB of gain compared to the classic SSCC and the non-adaptive UEC benchmarkers, while maintaining the same transmission bandwidth, throughput, latency, energy dissipation and decoding complexity.

REFERENCES

[1] J. Zou, H. Xiong, C. Li, R. Zhang and Z. He, "Lifetime and distortion optimization with joint source/channel rate adaptation and network coding-based error control in wireless video sensor Networks," *IEEE Trans. Veh. Technol.*, vol. 60, no. 3, pp. 1182–1194, 2011.

[2] Y. Huo, C. Zhu and L. Hanzo,, "Spatio-temporal iterative source-channel decoding aided video transmission," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1597–1609, 2013.

[3] N.S. Othman, M. El-Hajjar, O. Alamri, S. X. Ng and L. Hanzo, "Iterative AMR-WB source and channel decoding using differential space-time spreading-assisted sphere-packing modulation," *IEEE Trans. Veh. Technol.*, vol. 58, no. 1, pp. 484–490, 2009.

[4] *Advanced video coding for generic audiovisual services*. ITU-T Std. H.264, March 2005.

[5] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. 21, no. 2, pp. 194–203, March 1975.

[6] R. G. Maunder, W. Zhang, T. Wang and L. Hanzo, "A unary error correction code for the near-capacity joint source and channel coding of symbol values from an infinite set," *IEEE Trans. Commun. (In Press)*. [Online]. Available: http://eprints.soton.ac.uk/341736/

[7] J. L. Massey, "Joint source and channel coding," in *Commun. Systems and Random Process Theory*, The Netherlands: sijthofoo and Noordhoff, Dec. 1978, pp. 279–293.

[8] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate(Corresp.)," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[9] X. Jaspar and L. Vandendorpe, "Performance and convergence analysis of joint source-channel turbo schemes with variable length codes," *Acoustics, Speech, and Signal Processing, (ICASSP '05). IEEE International Conference on*, vol. 3, no. , pp. 485–488, Mar. 2005.

[10] M. Tüchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, Taipei, Taiwan, Nov. 2002, pp. 1358–1362.

[11] F. Brannstrom, L. K. Rasmussen and A. J. Grant, "Convergence analysis and optimal scheduling for multiple concatenated codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3354–3364, Sep. 2005.

[12] N. L. Johnson, A. W. Kemp and S. Kotz, *Univariate Discrete Distributions*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2005.

[13] T. Wang, R. G. Maunder and L. Hanzo, "Near-capacity joint source and channel coding of symbol values from an infinite source set using elias gamma error correction codes," *IEEE Trans. Commun. (Accepted)*, 2012. [Online]. Available: http://eprints.soton.ac.uk/344059/

[14] R. G. Maunder, W. Zhang, T. Wang and L. Hanzo,, "Derivations for a unary error correction code for the near-capacity joint source and channel coding of symbol values from an innite set." [Online]. Available: http://eprints.soton.ac.uk/341736/

[15] L. Hanzo, R. G. Maunder, J. Wang, and L.-L. Yang, *Near-Capacity Variable Length Coding*. Chichester, UK: Wiley, 2010. [Online]. Available: http://eprints.ecs.soton.ac.uk/20911/.

[16] R.G. Maunder and L. Hanzo, "Extrinsic information transfer analysis and design of block-Based intermediate codes," *IEEE Trans. Veh. Technol.*, vol. 60, no. 3, pp. 762–770, Mar. 2011.

[17] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[18] Nasruminallah and L. Hanzo,, "EXIT-chart optimized short block codes for iterative joint source and channel decoding in H.264 video telephony," *IEEE Trans. Veh. Technol.*, vol. 58, no. 8, pp. 4306–1315, 2009.

[19] S. ten Brink, "Convergence of iterative decoding," *Electron. Lett.*, vol. 35, no. 10, pp. 806–808, May 1999.

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

[21] J. Kliewer, N. Goertz, and A. Mertins, "Iterative source-channel decoding with Markov random field source models," *IEEE Trans. Signal Processing*, vol. 54, no. 10, pp. 3688–3701, Oct. 2006.

[22] S. ten Brink, G. Kramer and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[23] I. Land, P. Hoeher, and S. Gligorevic, "Computation of symbol-wise mutual information in transmission systems with logAPP decoders and application to EXIT charts," in *Proc. International ITG Conf. Source and Channel Coding (SCC)*, Germany, Jan. 2004, pp. 195–202.

[24] A. Ashikhmin, G. Kramer and S. ten Brink, "Code rate and the area under extrinsic information transfer curves," in *Proc. Inf. Theory, 2002 IEEE International Symposium on*, Lausanne, Switzerland, Jun. 2002, p. 115.

[25] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.

[26] H. Chen, R.G. Maunder and L. Hanzo, "Low-Complexity Multiple-Component Turbo-Decoding-Aided Hybrid ARQ," *IEEE Trans. Veh. Technol.*, vol. 60, no. 4, pp. 1571–1577, 2011.

[27] L. Li, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo, "A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks," *Very Large Scale Integration (VLSI) Systems, IEEE Trans. on*, vol. PP, no. 99, pp. 1–9, 2011.

[28] J. Hagenauer, "The EXIT chart Introduction to extrinsic information transfer in iterative decoding," in *Proc. of the European Sign. Proc. Conf*, Vienna, Austria, Sep. 2004, pp. 1541–1548.

[29] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), TS 36.212, Sept. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36212.htm

[30] L. Hanzo, T. H. Liew, B. L. Yeap, R. Y. S. Tee, S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding*. John Wiley & Sons, 2011. [Online]. Available: http://eprints.soton.ac.uk/258252/.