Near-Capacity Joint Source and Channel Coding of Symbol Values from an Infinite Source Set Using Elias Gamma Error Correction Codes

Tao Wang, Wenbo Zhang, Robert G. Maunder, and, Lajos Hanzo

***Abstract*—In this paper we propose a novel low-complexity Joint Source and Channel Code (JSCC), which we refer to as the Elias Gamma Error Correction (EGEC) code. Like the recently-proposed Unary Error Correction (UEC) code, this facilitates the practical near-capacity transmission of symbol values that are randomly selected from a set having an infinite cardinality, such as the set of all positive integers. However, in contrast to the UEC code, our EGEC code is a universal code, facilitating the transmission of symbol values that are randomly selected using any monotonic probability distribution. When the source symbols obey a particular zeta probability distribution, our EGEC scheme is shown to offer a 3.4 dB gain over a UEC benchmarker, when Quaternary Phase Shift Keying (QPSK) modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. In the case of another zeta probability distribution, our EGEC scheme offers a 1.9 dB gain over a Separate Source and Channel Coding (SSCC) benchmarker.**

***Index Terms*—Source coding, channel coding, channel capacity, iterative decoding.**

I. Introduction

SHANNON'S source-channel separation theorem [1] states that near-capacity communication is possible, when employing Separate Source and Channel Coding (SSCC). For example, this may be achieved by combining a near-entropy source code, such as an adaptive arithmetic code [2] or a Lempel-Ziv code [3], with a near-capacity channel code, such as a Low Density Parity Check (LDPC) code [4] or a turbo code [5]. However, the source-channel separation theorem relies upon a number of assumptions, which may not be valid in practice [6]. For example, near-entropy adaptive arithmetic coding or Lempel-Ziv coding requires both the transmitter and receiver to accurately estimate the occurrence probability of every value that is adopted by the symbols that the source produces. However, the occurrence probability of rare symbol values cannot be accurately estimated until a sufficiently high number of symbols have been generated, imposing an excessive latency which cannot be tolerated in many practical applications. This problem becomes particularly severe, when the symbol values are selected from a set having an infinite cardinality, such as the set of all positive integers. Furthermore, transmission errors may cause the estimated symbol probabilities to become desynchronized between the transmitter and receiver, potentially causing the excessive propagation of decoding errors. These issues motivate the design of universal codes, such as the Elias Gamma (EG) code [7]. These codes facilitate the communication of symbols selected from infinite sets, without requiring any knowledge of the corresponding occurrence probabilities at either the transmitter or receiver. Other examples of universal codes include the Elias delta code [7], the Elias omega code [7], the Even-Rodeh code [8], the Stout code [9] and the Fibonacci code [10]. Furthermore, the exponential-Golomb code [11] is a parametrized universal code, which has the EG code as a special case. Universal codes are typically employed in multimedia codecs such as the H.264 video codec [11], where they are employed for encoding the values of various symbols, such as motion vectors. Our previous work [12, Figure 1] demonstrated that in H.264, these symbol values are selected from a set having a cardinality of approximately 1000. We also showed that these symbol values obey Zipf's law and may therefore be represented using a zeta probability distribution [13]. However, some residual redundancy remains in the source-coded bit-stream when EG codes are employed for representing symbols that are randomly selected from a zeta probability distribution, hence imposing a certain capacity loss and preventing near-capacity operation when SSCC is employed [12]. Furthermore, SSCC is sensitive to transmission errors, with a single bit error potentially causing the corruption of several video frames in H.264, for example.

Motivated by this, various Joint Source and Channel Codes (JSCCs) have been proposed for mitigating the impact of transmission errors, as well as mitigating the capacity loss that is imposed by residual redundancy, when employed for representing symbols values that are selected from a set having a *low* cardinality. However, all previous JSCCs suffer from an excessive decoding complexity, when the cardinality of the symbol value set is large, leading to an infinite complexity, when the cardinality is infinite [12]. For example, the complexity of Variable Length Error Correction (VLEC) codes was characterized in [14] and found to increase rapidly with the cardinality of the symbol value set. This motivated us to propose the Unary Error Correction (UEC) code [12], which is the first JSCC that has a low decoding complexity, when employed to represent symbols values that are selected from a

Manuscript received April 25, 2013; revised September 22 and November 19, 2013. The editor coordinating the review of this paper and approving it for publication was V. Stankovic.

The authors are with Electronics and Computer Science, University of Southampton, Hampshire, SO17 1BJ, UK (e-mail: {tw08r, wz4g11, rm, lh}@ecs.soton.ac.uk).

The financial support of the EPSRC, Swindon UK under the grants EP/J015520/1 and EP/L010550/1, that of the RCUK under the India-UK Advanced Technology Centre (IU-ARC) as well as of the EU under the CONCERTO project is gratefully acknowledged.

Digital Object Identifier 10.1109/TCOMM.2013.120213.130301

set having an infinite cardinality. When the channel's Signal to Noise Ratio (SNR) is sufficiently high, our UEC code facilitates reliable communication, without requiring any knowledge of the symbol value occurrence probabilities at either the transmitter or receiver. However, once the UEC decoder has recovered a relatively-low number of source symbols, the receiver may estimate the occurrence probabilities of the most frequently occurring symbol values. This knowledge may then be employed to exploit the remaining residual redundancy for error correction, hence facilitating reliable communication at near-capacity SNRs. However, the UEC code is based on the unary code [15], which is not a universal code[1]. This limits its employment to situations where the symbol values have only particular probability distributions, including only a limited subset of the zeta probability distributions. Therefore, the employment of the UEC code is prevented in the case of arbitrary probability distributions, since its average codeword length may become infinite in these cases. Furthermore, for some zeta probability distributions that are supported by the UEC code, it may be outperformed by a corresponding SSCC benchmarker, even though the latter imposes capacity loss [16].

Against this background, this paper proposes a *universal* JSCC for the near-capacity transmission of infinite-cardinality symbol alphabets that are randomly selected using *any* arbitrary monotonic probability distribution. Owing to this, the proposed JSCC has a much wider applicability than the UEC of [12], facilitating its employment for the entire set of zeta probability distributions. Like the UEC code of [12], the proposed JSCC does not require any knowledge of the symbol occurrence probabilities at either the transmitter or receiver, when the channel SNR is sufficiently high. However, once the receiver has estimated the occurrence probabilities of the most frequently occurring symbol values, reliable communication at near-capacity SNRs is facilitated. Rather than employing a unary code as its basis, the proposed code is based upon the universal EG code, hence we refer to it as the Elias Gamma Error Correction (EGEC) code. As described in Section II, the EGEC encoder decomposes each input symbol into two sub-symbols, which are encoded separately by two distinct sub-encoders. The first sub-encoder is referred to as the EGEC(UEC) encoder, which operates in the same manner as the UEC encoder of [12]. The second sub-encoder employs a serial concatenation of a Fixed length Code (FLC) and a Convolutional Code (CC) encoders, which we refer to as the EGEC(FLC-CC) encoder. As described in Section III, the EGEC decoder has corresponding sub-decoders, which operate on the basis of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [17] and the Soft Bit Source Decoding (SBSD) algorithm [18]. In Section IV, we propose a Unequal Error Protection (UEP) scheme for optimizing the relative contribution of the two sub-codes to the encoding process, facilitating near-capacity operation at a low decoder complexity. We will demonstrate in Section V that if the source symbols obey a particular zeta probability distribution, our EGEC scheme offers a 3.4 dB gain over a UEC benchmarker, when Quaternary Phase Shift Keying (QPSK) is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. For another zeta probability distribution, our EGEC scheme will be shown to offer a 1.9 dB gain over a SSCC benchmarker, which we refer to as the Elias Gamma and Convolutional Code (EG-CC) scheme. Additionally, we will consider a wide range of other zeta probability distributions and will show that our EGEC scheme is capable of offering gains over the relevant benchmarkers in each case. Finally, we offer our conclusions in Section VI.

[1]A universal code is a countably infinite prefix code set. When encoding a symbol set following any monotonic probability distribution, the average codeword length is bounded by a function of the entropy of the distribution [7].

II. EGEC Encoder

In this section, we introduce the EGEC encoder, which is illustrated in Figure 1. In Section II-A, we discuss the motivation for decomposing the input symbols into two sub-symbols and describe the operation of the corresponding symbol splitter in Figure 1, which is labeled S. The operation of the EGEC(UEC) and EGEC(FLC-CC) encoders is described in Sections II-B and II-C. Finally, Section II-D describes the serial concatenation of the EGEC encoder with the Unity Rate Code (URC) encoder and QPSK modulator of Figure 1.

A. Decomposition of symbols into pairs of sub-symbols

As shown in Figure 1, the EGEC encoder is designed for representing a vector $\mathbf{d} = [d_i]_{i=1}^{a}$ comprising a number of symbols, which can be obtained as a realization of a corresponding vector $\mathbf{D} = [D_i]_{i=1}^{a}$ comprising a number of Independent and Identically Distributed (IID) Random Variables (RVs). Each RV D_i adopts the symbol value $d \in \mathbb{N}_1$ with probability $\Pr(D_i = d) = P(d)$, where $\mathbb{N}_1 = \{1, 2, 3, \ldots\}$ is the infinite-cardinality source set comprising all positive integers.

In this paper, we focus our attention on symbol values that are randomly selected from a zeta distribution [13], since the parameters of multimedia codecs typically obey Zipf's law [12, Figure 1]. The zeta distribution is defined as

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

where $\zeta(s)=\sum_{d\in\mathbb{N}_1} d^{-s}$ is the Riemann zeta function and $s > 1$. In this case, $p_1 = \Pr(D_i = 1) = 1/\zeta(s)$ and the symbol entropy is given by

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

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

As shown in Table I, source encoders such as unary or EG encoders represent each symbol d_i in the vector $\mathbf{d}$ using a corresponding binary codeword, namely $\mathrm{Unary}(d_i)$ or $\mathrm{EG}(d_i)$, respectively. Note that for the convenience of our ensuing discussions, the unary codewords shown in Table I are the complements of those that are conventionally employed, for example in [12, Table I]. The average codeword length is given by

$$l = \sum_{d\in\mathbb{N}_1} P(d)l(d), \tag{3}$$

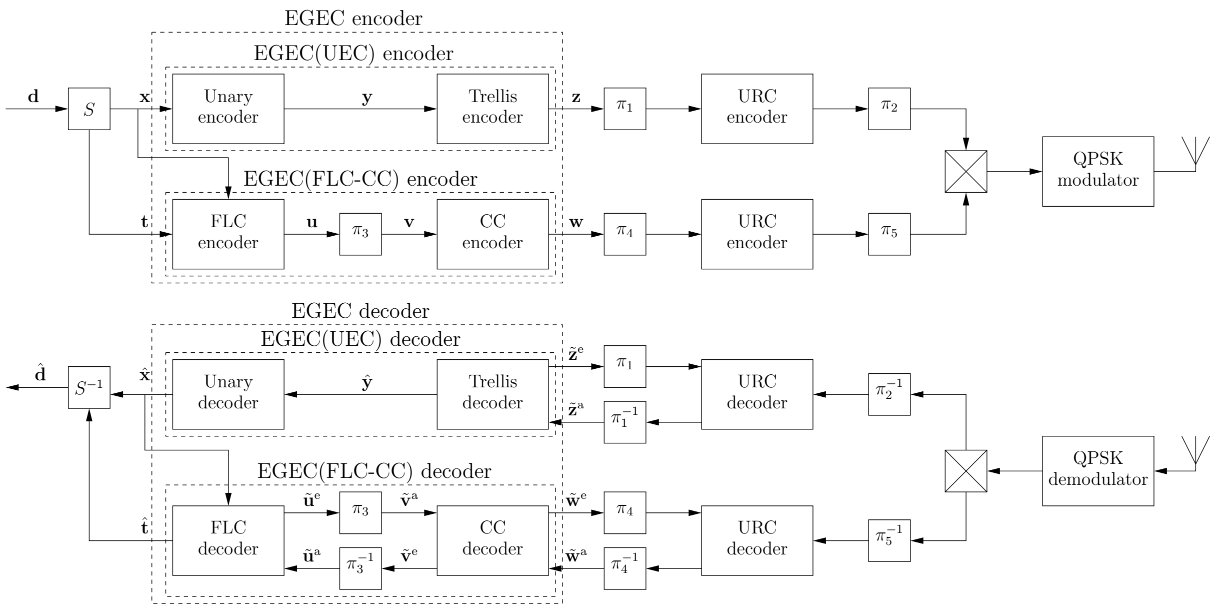


Fig. 1. Schematic of the EGEC code, when serially concatenated with URC and Gray-coded QPSK modulation schemes. Bold notation without a diacritic is used to denote a symbol vector or a bit vector. A diacritical hat represents a reconstruction of the symbol or bit vector having the corresponding notation. A diacritical tilde represents an LLR vector pertaining to the bit vector with the corresponding notation. A roman superscript 'a' is employed to denote an *a priori* LLR vector, while 'e' is employed for extrinsic LLR vectors. Furthermore, $\{\pi_1, \ldots, \pi_5\}$ represent interleavers, while $\{\pi_1^{-1}, \ldots, \pi_5^{-1}\}$ represent the corresponding deinterleavers. Puncturing may also be performed in π_2 and π_5, while the corresponding depuncturing operations take place in π_2^{-1} and π_5^{-1}. Multiplexing and demultiplexing is performed in the crossed boxes.

TABLE I
THE FIRST TWELVE CODEWORDS OF VARIOUS SOURCE CODES.

d_i	Unary(d_i)	EG(d_i)	x_i	Unary(x_i)	t_i	FLC(t_i)
1	1	1	1	1	0	
2	01	010	2	01	0	0
3	001	011	2	01	1	1
4	0001	00100	3	001	0	00
5	00001	00101	3	001	1	01
6	000001	00110	3	001	2	10
7	0000001	00111	3	001	3	11
8	00000001	0001000	4	0001	0	000
9	000000001	0001001	4	0001	1	001
10	0000000001	0001010	4	0001	2	010
11	00000000001	0001011	4	0001	3	011
12	000000000001	0001100	4	0001	4	100

where $l(d)$ is the length of the d^{th} codeword.

In the case of a unary code, the length of the codeword $\text{Unary}(d_i)$ is yielded by $l(d_i) = d_i$, giving an average codeword length of

$$l_{\text{Unary}(d_i)} = \frac{\zeta(s-1)}{\zeta(s)}, \quad (4)$$

when the source symbols obey the zeta distribution of (1). However, the average unary codeword length l is only finite for $s > 2$ and hence for $p_1 > 0.608$. Despite this, we proposed a JSCC scheme based on the unary code in [12], since its codewords have a relatively simple structure, which can be readily exploited for error correction. More specifically, the structure of the unary codewords can be described by the UEC trellis of [12], without requiring an infinite number of trellis transitions and states.

By contrast, an EG codeword $\text{EG}(d_i)$ has a length of $l(d_i) = 2\lfloor \log_2(d_i) \rfloor + 1$. When the source symbols obey the zeta distribution, the average codeword length becomes

$$l_{\text{EG}(d_i)} = 1 - \frac{2\zeta'(s)}{\ln(2)\zeta(s)} - \frac{2}{\zeta(s)} \sum_{x \in \mathbb{N}_1} x^{-s} \text{frac}(\log_2(x)), \quad (5)$$

where the $\text{frac}(\cdot)$ operator yields the fractional part of the operand, as in $\text{frac}(3.4) = 0.4$. Note that the average EG codeword length l is finite for all zeta distributions, not just those for which $p_1 > 0.608$.

In this paper, we develop a trellis representation of the EG code by observing that the codeword $\text{EG}(d_i)$ is prefixed by a unary codeword $\text{Unary}(x_i)$, where we have [7]

$$x_i = \lfloor \log_2(d_i) \rfloor + 1, \quad (6)$$

as may be observed in Table I. Furthermore, the length of the EG codeword's remaining suffix $\text{FLC}(t_i)$ depends on the selected unary codeword $\text{Unary}(x_i)$. More specifically, the suffix $\text{FLC}(t_i)$ comprises $(x_i - 1)$ bits, which form the binary representation of the decimal value t_i, where

$$t_i = d_i - 2^{\lfloor \log_2(d_i) \rfloor}. \quad (7)$$

Inspired by this, the splitter S of Figure 1 decomposes each symbol d_i in the vector $\mathbf{d}$ into two sub-symbols, namely into x_i and t_i according to (6) and (7), where

$$d_i = 2^{x_i - 1} + t_i. \quad (8)$$

Each set of sub-symbols is concatenated to form the vectors $\mathbf{x} = \{x_i\}_{i=1}^{a}$ and $\mathbf{t} = \{t_i\}_{i=1}^{a}$. For example, the vector $\mathbf{d} = [6, 15, 1, 17, 2, 1, 1, 2]$ of $a = 8$ symbols yields the vector $\mathbf{x} = [3, 4, 1, 5, 2, 1, 1, 2]$ of $a = 8$ sub-symbols and the vector $\mathbf{t} = [2, 7, 0, 1, 0, 0, 0, 0]$ comprising $a = 8$ sub-symbols.

As shown in Figure 1, each sub-symbol x_i in the vector $\mathbf{x}$ is encoded by the EGEC(UEC) encoder of Section II-B, while each sub-symbol t_i in the vector $\mathbf{t}$ is encoded by the

EGEC(FLC-CC) encoder of Section II-C, in order to produce the codewords Unary(x_i) and FLC(t_i), as exemplified in Table I, respectively. Note that since the codewords Unary(x_i) and FLC(t_i) collectively comprise the same number of bits as the codeword EG(d_i), the proposed EGEC code produces the same number of unary- and FLC-encoded bits as are produced by an EG encoder. Therefore, since an EG code is a universal code, so too is the proposed EGEC code, granting it a finite average codeword length when the symbol values are selected according to any monotonic probability distribution.

B. EGEC(UEC) encoder

As shown in Figure 1, the vector $\mathbf{x} = [x_i]_{i=1}^a$ is encoded by the EGEC(UEC) encoder, which operates in the same way as the UEC encoder of [12]. The vector of sub-symbols $\mathbf{x}$ can be modeled as a realization of a vector of RVs $\mathbf{X} = [X_i]_{i=1}^a$, where each RV X_i adopts a symbol value from the set $x \in \mathbb{N}_1$ with a probability of $\Pr(X_i = x) = P(x)$. In the scenario where the RV D_i obeys the zeta distribution of (1), the RV X_i will obey the distribution

$$P(x) = \frac{1}{\zeta(s)} \sum_{d=2^{x-1}}^{2^x-1} d^{-s}, \tag{9}$$

where the entropy of the RV X_i is given by

$$H_X = \log_2[\zeta(s)] - \frac{1}{\zeta(s)} \sum_{x\in\mathbb{N}_1} \left(\sum_{d=2^{x-1}}^{2^x-1} d^{-s} \right) \log_2 \left(\sum_{d=2^{x-1}}^{2^x-1} d^{-s} \right), \tag{10}$$

The sub-symbol vector $\mathbf{x}$ is forwarded to the unary encoder of Figure 1, which represents each symbol x_i in the vector using the corresponding x_i-bit unary codeword $\text{Unary}(x_i)$ of Table I. When the sub-symbols in the vector $\mathbf{x}$ obey the zeta distribution of (9), the average unary codeword length is given by $l_1 = l_{\text{Unary}(x_i)}$, where we have

$$\begin{aligned} l_1 &= \sum_{x\in\mathbb{N}} P(x) \cdot x \\ &= 1 - \frac{\zeta'(s)}{\ln(2)\zeta(s)} - \frac{1}{\zeta(s)} \sum_{x\in\mathbb{N}_1} x^{-s} \text{frac}[\log_2(x)], \end{aligned} \tag{11}$$

which is guaranteed to be finite, regardless of the value of $s > 1$. Note that (11) may be derived from (5) by observing that $l_{\text{Unary}(x_i)} = (l_{\text{EG}(d_i)} + 1)/2$, as shown in Table I. Following this, the unary codewords are concatenated for generating the b-bit vector $\mathbf{y} = [y_j]_{j=1}^b$ of Figure 1. For example, the vector $\mathbf{x} = [3, 4, 1, 5, 2, 1, 1, 2]$ of $a = 8$ sub-symbols is represented by the vector $\mathbf{y} = 0010001100001011101$ of $b = 19$ bits.

As shown in Figure 1, the vector of concatenated unary codewords $\mathbf{y}$ is input to the trellis encoder of [12]. This operates on the basis of a UEC trellis, such as the $r_1 = 4$-state trellis that is exemplified in Figure 2. This represents a special case of the generalized r_1-state UEC trellis of [12, Figure 3(a)], albeit with complemented values for y_j owing to the inversion of the unary codewords of Table I relative to those of [12, Table I]. 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 $m_{j-1} \in \{1, 2, \ldots, r_1\}$ to its next state $m_j \in \{1, 2, \ldots, r_1\}$, 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

$$m_j = \begin{cases} 1 + \text{odd}(m_{j-1}) & \text{if } y_j = 1 \\ \min[m_{j-1} + 2, r_1 - \text{odd}(m_{j-1})] & \text{if } y_j = 0 \end{cases}, \tag{12}$$

where the number of possible states r_1 has to be even and the encoding process always begins from the state $m_0 = 1$. The function $\text{odd}(\cdot)$ 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 $\mathbf{m} = [m_j]_{j=0}^b$ comprising $(b + 1)$ state values. For example, the bit vector $\mathbf{y} = 0010001100001011101$ yields the path $\mathbf{m} = [1, 3, 3, 2, 4, 4, 4, 1, 2, 4, 4, 4, 4, 1, 3, 2, 1, 2, 4, 1]$ through the $r_1 = 4$-state trellis of Figure 2. Following this, the trellis encoder represents each bit y_j in the vector $\mathbf{y}$ by an n_1-bit codeword $\mathbf{z}_j$. This is selected from the set of $r_1/2$ codewords $\mathbb{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{r_1/2}\}$ or from the complementary set $\overline{\mathbb{C}} = \{\overline{\mathbf{c}_1}, \overline{\mathbf{c}_2}, \ldots, \overline{\mathbf{c}_{r_1/2}}\}$, which is achieved according to

$$\mathbf{z}_j = \begin{cases} \overline{\mathbf{c}_{\lceil m_{j-1}/2 \rceil}} & \text{if } y_j \neq \text{odd}(m_{j-1}) \\ \mathbf{c}_{\lceil m_{j-1}/2 \rceil} & \text{if } y_j = \text{odd}(m_{j-1}) \end{cases}. \tag{13}$$

Following this, the selected codewords are concatenated to obtain the bn_1-bit vector $\mathbf{z} = [z_k]_{k=1}^{bn_1}$ of Figure 1. For example, the vector $\mathbf{y} = 0010001100001011101$ of $b = 19$ bits is represented by the vector $\mathbf{z} = 10001101111110001011111111001011100010100$ of $bn_1 = 38$ bits, when employing the UEC trellis of Figure 2, which has $r_1 = 4$ states and the $n_1 = 2$-bit codewords $\mathbb{C} = \{01, 11\}$. Note that the selection of the parameter r_1 is discussed in Section IV.

Note that UEC trellis encoder operates in a similar manner to a CC encoder, but with some important differences, as follows.

1) The UEC trellis encoder is specifically designed for maintaining synchronization with the unary codewords that are concatenated to form the bit vector $\mathbf{y}$. More

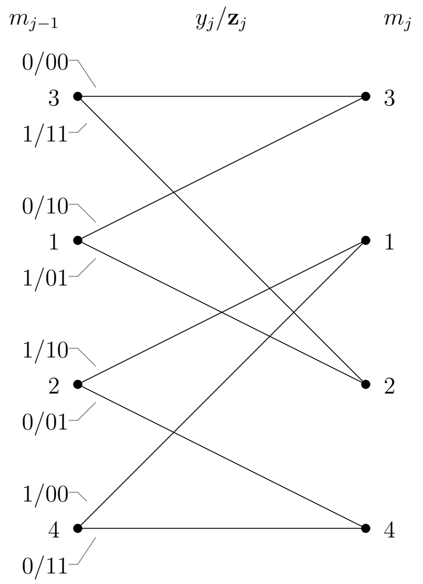


Fig. 2. An $r_1 = 4$-state $n_1 = 2$-bit UEC trellis, where $\mathbb{C} = \{01, 11\}$.

specifically, the last bit y_j in each unary codeword $\text{Unary}(x_i)$ is guaranteed to induce a transition in the state $m_j = 1$ or state $m_j = 2$, depending on whether the corresponding symbol x_i has an odd or even index i. This is exploited by the UEC trellis decoder in order to mitigate capacity loss, as described in Section III-A. By contrast, in a generalized CC encoder, the last bit in each unary codeword can potentially cause a transition into any state, preventing synchronization.

2) The unary encoded bit vector $\mathbf{y}$ is guaranteed to terminate the UEC trellis into state $m_b = 1$ or state $m_b = 2$, depending on whether the length a of the symbol vector $\mathbf{x}$ is odd or even. This may be exploited by the UEC trellis decoder in order to assist its operation, as described in Section III-A. By contrast, a generalized CC encoder is not terminated by the unary encoded bit vector $\mathbf{y}$.
3) The UEC trellis is designed to obey symmetry and to rely on complementary codewords, so that the binary values in the vector $\mathbf{z}$ are equiprobable. As described in [12], this is a necessary condition for avoiding capacity loss. By contrast, CC encoders produce binary values that are not guaranteed to be equiprobable, unless they are specifically parametrized for this purpose.

Since the binary values in the vector $\mathbf{z}$ are equiprobable, the average coding rate of the EGEC(UEC) encoder is given by

$$R_1^{\text{o}} = \frac{H_X}{l_1 n_1}. \tag{14}$$

Here, we employ the roman superscript 'o' to indicate that this coding rate relates to the outer code of a serial concatenation, namely the EGEC(UEC) code shown in Figure 1.

C. EGEC(FLC-CC) encoder

As shown in Figure 1, the EGEC(FLC-CC) encoder requires both $\mathbf{t}$ and $\mathbf{x}$ vectors, in order to perform FLC encoding. As described in Section II-A, this is because each sub-symbol t_i in the vector $\mathbf{t}$ is mapped to an FLC codeword $\text{FLC}(t_i)$, having the length $(x_i - 1)$, where x_i is the corresponding sub-symbol in the vector $\mathbf{x}$. When the sub-symbols in the vector $\mathbf{x}$ obey the distribution of (9), the average FLC codeword length is given by $l_2 = l_{\text{FLC}(t_i)}$, where we have

$$\begin{aligned} l_2 &= \sum_{x\in\mathbb{N}} P(x)\cdot(x-1) \\ &= -\frac{\zeta'(s)}{\ln(2)\zeta(s)} - \frac{1}{\zeta(s)}\sum_{x\in\mathbb{N}_1} x^{-s}\text{frac}[\log_2(x)], \end{aligned} \tag{15}$$

which is guaranteed to be finite, regardless of the value of $s > 1$.

Owing to the dependencies between $\mathbf{t}$ and $\mathbf{x}$, we model the sub-symbol vector $\mathbf{t}$ as a realization of a RV vector $\mathbf{T} = [T_i]_{i=1}^{a}$, where each RV T_i is dependent on the corresponding RV X_i. By considering (1) and (8), the joint probability $\Pr(T_i = t, X_i = x) = P(t, x)$ is given by

$$P(t,x) = \frac{1}{\zeta(s)}(2^{x-1}+t)^{-s}, \tag{16}$$

where $0 \leq t < 2^{x-1}$. Furthermore, the conditional probability $\Pr(T_i = t | X_i = x) = P(t|x)$ is given by

$$P(t|x) = \frac{P(t,x)}{P(x)} = \frac{(2^{x-1}+t)^{-s}}{\sum\limits_{d=2^{x-1}}^{2^x-1} d^{-s}}, \tag{17}$$

where $0 \leq t < 2^{x-1}$. Finally, the conditional entropy of the RV T_i is given by

$$H_{T|X} = \sum_{x\in\mathbb{N}_1}\sum_{t=0}^{2^{x-1}-1} P(t,x)\log_2\left(\frac{1}{P(t|x)}\right). \tag{18}$$

The codewords $\text{FLC}(t_i)$ are concatenated to obtain the $(b-a)$-bit vector $\mathbf{u} = [u_e]_{e=1}^{b-a}$ of Figure 1. For the example of $\mathbf{x} = [3,4,1,5,2,1,1,2]$ and $\mathbf{t} = [2,7,0,1,0,0,0,0]$, the combination $x_1 = 3$ and $t_1 = 2$ yields the codeword $\text{FLC}(t_1) = 10$, as shown in Table I. Similarly, $x_2 = 4$ and $t_2 = 7$ yields $\text{FLC}(t_2) = 111$, while $x_3 = 1$ and $t_3 = 0$ yields an empty bit vector for $\text{FLC}(t_3)$. Completing this encoding process and concatenating the resultant codewords yields the vector $\mathbf{u} = 10111000100$ of $(b-a) = 11$ bits. The bit vector $\mathbf{u}$ is interleaved in the block π_3 of Figure 1, in order to obtain the vector $\mathbf{v} = [v_e]_{e=1}^{b-a}$. Note that the binary values in the vectors $\mathbf{u}$ and $\mathbf{v}$ will not be equiprobable in general.

This motivates the employment of the r_2-state n_2-bit recursive CC encoders of [12, Table II], since these produce equiprobable binary values for the encoded bit vector $\mathbf{w} = [w_f]_{f=1}^{n_2(b-a)}$ of Figure 1. This is necessary because producing equiprobable binary values is a necessary condition for avoiding capacity loss [12]. For example, if the $r_2 = 4$-state $n_2 = 2$-bit recursive CC encoder of [12, Table II] is employed for encoding the $(b-a) = 11$-bit vector $\mathbf{v} = 00101100110$, the $n_2(b-a) = 22$-bit vector of $\mathbf{w} = 0000111000101011001010$ is generated. The average coding rate of the EGEC(FLC-CC) encoder is given by

$$R_2^{\text{o}} = \frac{H_{T|X}}{l_2 n_2}. \tag{19}$$

D. Integration of the EGEC encoder into a transmitter

Following EGEC encoding, the bit vectors $\mathbf{z}$ and $\mathbf{w}$ are interleaved by π_1 and π_4, URC encoded using accumulators [19] and then interleaved again by π_2 and π_5, as shown in Figure 1. These URC encoders are recursive and have a coding rate of unity, satisfying the corresponding conditions that are sufficient for facilitating near-capacity operation [20], [21]. Alternatively, LDPC or turbo codes may be employed for this purpose, although this implies a significantly increased complexity. This is because turbo decoders employ the iterative operation of two component decoders, while LDPC decoders employ the iterative operation of variable nodes and check nodes. By contrast, URC codes comprise only a single component, requiring no internal iterations. Note that the lengths required for these interleavers and URC codes depend on the particular sub-symbol values in the vector $\mathbf{x}$. However, these components can glean the required lengths from the lengths of their respective input bit vectors. Puncturing may also be performed within π_2 and π_5, in order to achieve the desired throughput for the transmitter, as well as for UEP, as discussed in Section IV. While interleaving and URC encoding are associated with one input bit per output bit, puncturing within π_2 and π_5 are respectively associated with coding rates of $R_1^{\mathrm{i}} \geq 1$ and $R_2^{\mathrm{i}} \geq 1$ input bits per output bit. Here, we employ the roman superscript 'i' to indicate that these coding rates relate to the inner codes of serial concatenations, namely the two URC codes shown in Figure 1. Following URC encoding, the multiplexer of Figure 1 appends the encoded bit sequence derived from the EGEC(FLC-CC) encoder onto the end of that derived from the EGEC(UEC) encoder. Following this, $M = 4$-ary Gray-coded QPSK modulation may be employed for transmission, as shown in Figure 1. Note that other mapping schemes or a modulation scheme having a higher order M can be employed instead, although this may increase the complexity of the receiver, as we will discuss in Section IV. The throughput of the transmitter is given by

$$\eta = \frac{H_D \log_2(M)}{l_1 n_1 / R_1^{\mathrm{i}} + l_2 n_2 / R_2^{\mathrm{i}}}. \qquad (20)$$

III. EGEC Decoder

In this section, we describe the operation of the EGEC decoder of Figure 1. The EGEC(UEC) decoder and EGEC(FLC-CC) decoder are described in Sections III-A and III-B, respectively. Following this, Section III-C discusses the serial concatenation of the EGEC decoder with the URC decoder and QPSK demodulator of Figure 1.

A. EGEC(UEC) decoder

As shown in Figure 1, the EGEC(UEC) decoder's trellis decoder is provided with a vector of *a priori* Logarithmic Likelihood Ratios (LLRs) $\tilde{\mathbf{z}}^{\mathrm{a}} = [\tilde{z}_k^{\mathrm{a}}]_{k=1}^{bn_1}$ that pertain to the corresponding bits in the vector $\mathbf{z}$. The trellis decoder operates on the basis of the BCJR algorithm of [12, Section IV-A]. This generates the vector of extrinsic LLRs $\tilde{\mathbf{z}}^{\mathrm{e}} = [\tilde{z}_k^{\mathrm{e}}]_{k=1}^{bn_1}$, which is provided for the next iteration of the concatenated URC decoder's operation. Following the completion of iterative decoding, the trellis decoder may also be employed to generate the vector of *a posteriori* LLRs $\tilde{\mathbf{y}}^{\mathrm{p}} = [\tilde{y}_j^{\mathrm{p}}]_{j=1}^{b}$ that pertain to the corresponding bits in the vector $\mathbf{y}$. The unary decoder of Figure 1 sorts the values in this LLR vector in order to identify the a number of bits in the vector $\mathbf{y}$ that are most likely to have values of one. A hard decision vector $\hat{\mathbf{y}} = [\hat{y}_j]_{j=1}^{b}$ is then obtained by setting the value of these a bits to one and the value of all other bits to zero. Here, the value of a is assumed to be perfectly known to the receiver and may be reliably conveyed by the transmitter using a small amount of side information in practice. Finally, the bit vector $\hat{\mathbf{y}}$ can be unary decoded in order to generate the sub-symbol vector $\hat{\mathbf{x}} = [\hat{x}_i]_{i=1}^{a}$ of Figure 1, which is guaranteed to comprise a number of sub-symbols. This has the benefit of mitigating, although not totally eliminating, the error propagation that may occur owing to the variable lengths of the unary codewords.

Note that the trellis decoder's BCJR algorithm has only a modest complexity, since it may employ a low number r_1 of states. Furthermore, it facilitates error correction even if the sub-symbol probability distribution $P(x)$ is unknown, provided that the channel SNR is sufficiently high, as we shall demonstrate in Section V. However, once a sufficient number of sub-symbol vectors $\hat{\mathbf{x}}$ has been recovered, the average unary codeword length l_1 and the sub-symbol probabilities $P(x)$ for $x \leq r_1/2 - 1$ may be heuristically estimated. When decoding subsequent sub-symbol vectors, this information may be exploited by the BCJR algorithm in order to facilitate error correction at near-capacity SNRs [12, Section V]. Note that this is made possible by the termination of the UEC trellis and its synchronization with the unary codewords [12, Section V], as described in Section II-B. Also note that the availability of l_1 and $P(x)$ for $x \leq r_1/2 - 1$ is assumed throughout the remainder of this paper, unless explicitly stated otherwise.

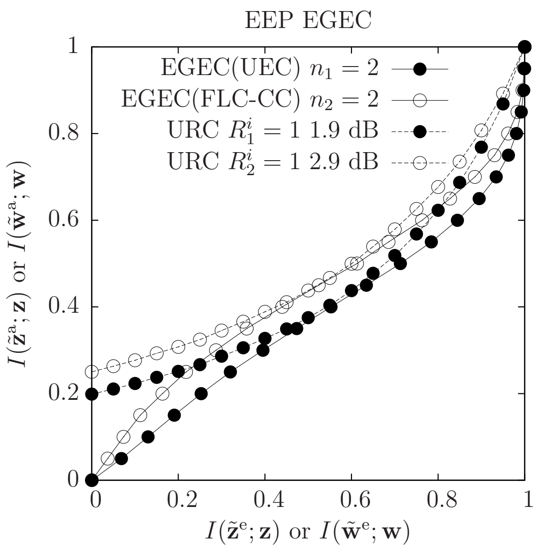


Fig. 3. EXIT charts of the EEP EGEC scheme. Here, the symbols of $\mathbf{d}$ obey a zeta distribution having $p_1 = 0.7967$ and the codewords comprise the numbers of bits n_1 and n_2. Furthermore, the punctured URC decoders adopt the coding rates R_1^{i} and R_1^{i}, for Gray-coded QPSK modulation onto an uncorrelated narrowband Rayleigh fading channel having various E_{b}/N_0. The EXIT curves are provided for an EGEC(UEC) code having $r_1 = 4$ states, as well as for an EGEC(FLC-CC) code having $x_{\max} = 3$.

The transformation of $\tilde{\mathbf{z}}^{\mathrm{a}}$ into $\tilde{\mathbf{z}}^{\mathrm{e}}$ may be characterized by plotting the inverted EGEC(UEC) EXtrinsic Information Transfer (EXIT) curve in an EXIT chart [22], as exemplified in Figure 3. Note that if UEC codewords comprising at least $n_1 = 2$ bits are employed, then the free distance d_{free} of the UEC code will be at least two, and its EXIT curve will reach the $(1, 1)$ point at the top right corner of the EXIT chart [23]. Reaching this point is important because in this case a vanishingly low Bit Error Ratio (BER) may be attained. In the scenario where the *a priori* LLR vector $\tilde{\mathbf{z}}^{\mathrm{a}}$ is modeled by the transmission of $\mathbf{z}$ over a Binary Erasure Channel (BEC), the EXIT chart area A_1^{o} that is situated below the inverted EGEC(UEC) EXIT curve of the demodulator is given by [12, (14)]

$$\begin{aligned} A_1^{\mathrm{o}} &= \tfrac{1}{l_1 n_1} \textstyle\sum_{x=1}^{\frac{r_1}{2}-1} H[P(x)] + \tfrac{2}{l_1 n_1} H\left[1 - \textstyle\sum_{x=1}^{\frac{r_1}{2}-1} P(x)\right] \\ &+ \tfrac{1}{l_1 n_1} H\left[l_1 - \tfrac{r_1}{2} + \textstyle\sum_{x=1}^{r_1/2-1} P(x)\left(\tfrac{r_1}{2} - x\right)\right] \\ &- \tfrac{1}{l_1 n_1} H\left[l_1 + 1 - \tfrac{r_1}{2} + \textstyle\sum_{x=1}^{r_1/2-1} P(x)\left(\tfrac{r_1}{2} - 1 - x\right)\right]. \end{aligned} \qquad (21)$$

B. EGEC(FLC-CC) decoder

As shown in Figure 1, the EGEC(FLC-CC) decoder iteratively exchanges extrinsic information between the CC decoder and the FLC decoder. More specifically, the n_2-bit r_2-state CC decoder employs the BCJR algorithm for transforming the *a priori* LLR vectors $\tilde{\mathbf{v}}^{\mathrm{a}} = [\tilde{v}^{\mathrm{a}}_e]_{e=1}^{b-a}$ and $\tilde{\mathbf{w}}^{\mathrm{a}} = [\tilde{w}^{\mathrm{a}}_f]_{f=1}^{n_2(b-a)}$ into the extrinsic LLR vectors $\tilde{\mathbf{v}}^{\mathrm{e}} = [\tilde{v}^{\mathrm{e}}_e]_{e=1}^{b-a}$ and $\tilde{\mathbf{w}}^{\mathrm{e}} = [\tilde{w}^{\mathrm{e}}_f]_{f=1}^{n_2(b-a)}$. Here, the extrinsic LLR vector $\tilde{\mathbf{w}}^{\mathrm{e}}$ is provided for the next iteration of the concatenated URC decoder's operation. Meanwhile, the extrinsic LLRs of $\tilde{\mathbf{v}}^{\mathrm{e}}$ are deinterleaved π_3^{-1}, in order to obtain the *a priori* LLR vector $\tilde{\mathbf{u}}^{\mathrm{a}} = [\tilde{u}^{\mathrm{a}}_e]_{e=1}^{b-a}$. As shown in Figure 1, the *a priori* LLR vector $\tilde{\mathbf{u}}^{\mathrm{a}}$ is then forwarded to the FLC decoder, together with the sub-symbol vector $\hat{\mathbf{x}}$, which is provided by the EGEC(UEC) decoder. The sub-symbols of $\hat{\mathbf{x}}$ are employed for partitioning the *a priori* LLR vector $\tilde{\mathbf{u}}^{\mathrm{a}}$ into sub-vectors, where the i^{th} sub-vector comprises $(\hat{x}_i - 1)$ bits. Note that since $\hat{\mathbf{x}}$ is guaranteed to contain a number of sub-symbols, the sum of the sub-vector lengths is given by $\sum_{i=1}^{a}(\hat{x}_i - 1)$, which is guaranteed to be equal to the length $(b-a)$ of the LLR vector $\tilde{\mathbf{u}}^{\mathrm{a}}$. The FLC decoder employs the SBSD algorithm of [18] to generate the vector of extrinsic LLRs $\tilde{\mathbf{u}}^{\mathrm{e}} = [\tilde{u}^{\mathrm{e}}_e]_{e=1}^{b-a}$. This is then interleaved in the block π_3 of Figure 1, in order to obtain the *a priori* LLR vector $\tilde{\mathbf{v}}^{\mathrm{a}}$ for the next iteration of the CC decoder's operation. Following the completion of iterative decoding, the FLC decoder may also be employed to generate the vector of a sub-symbols $\hat{\mathbf{t}} = [\hat{t}_i]_{i=1}^{a}$, as shown in Figure 1.

Note that the FLC decoder can recover the sub-symbol vector $\hat{\mathbf{t}}$ even if the conditional probabilities of (17) are unknown, provided that the channel SNR is sufficiently high, as we shall demonstrate in Section V. However, once a sufficient number of sub-symbol vectors $\hat{\mathbf{t}}$ has been recovered, the conditional probabilities $P(t|x)$ can be heuristically estimated for all pairs of t and x where $x \leq x_{\max}$. When decoding subsequent sub-symbol vectors, this information may be exploited by the SBSD algorithm in order to facilitate error correction at near-capacity SNRs [12, Section V]. More specifically, the SBSD algorithm can apply the conditional probabilities of (17) to each of the a sub-vectors of $\tilde{\mathbf{u}}^{\mathrm{a}}$ for which the corresponding sub-symbols in $\hat{\mathbf{x}}$ do not exceed $x_{\max}$. This improves the reconstruction of the corresponding sub-symbols in $\hat{\mathbf{t}}$, as well as providing extrinsic information for the corresponding LLRs in $\tilde{\mathbf{u}}^{\mathrm{e}}$. Note that zero values are adopted by the LLRs in $\tilde{\mathbf{u}}^{\mathrm{e}}$ which correspond to sub-symbols in $\hat{\mathbf{x}}$ that do exceed $x_{\max}$. Also note that the availability of the conditional probabilities $P(t|x)$ for $x \leq x_{\max}$ is assumed throughout the remainder of this paper, unless explicitly stated otherwise.

The transformation of $\tilde{\mathbf{w}}^{\mathrm{a}}$ into $\tilde{\mathbf{w}}^{\mathrm{e}}$ using the iterative operation of the CC and FLC decoders may be characterized by plotting the inverted EGEC(FLC-CC) EXIT curve in an EXIT chart [22]. This is exemplified in Figure 3 for the scenario where the CC and FLC decoders are operated, until iterative decoding convergence is achieved. Note that if CC codewords comprising at least $n_2 = 2$ bits are employed, then the free distance d_{free} of the CC code will be at least two, and the EGEC(FLC-CC) EXIT curve will reach the $(1, 1)$ point at the top right corner of the EXIT chart [23]. In the case where the *a priori* LLR vector $\tilde{\mathbf{w}}^{\mathrm{a}}$ is modeled by the transmission of $\mathbf{w}$ over a BEC, the EXIT chart area that is situated below the inverted EGEC(FLC-CC) EXIT curve is given by

$$\begin{aligned} A_2^{\mathrm{o}} &= \tfrac{1}{n_2} \textstyle\sum_{x=2}^{x_{\max}} \sum_{t=0}^{2^{x-1}-1} \tfrac{H[P(t|x)]}{x-1} \tfrac{(x-1)P(x)}{l_2} \\ &+ \tfrac{1}{n_2} \left(1 - \textstyle\sum_{x=2}^{x_{\max}} \sum_{t=0}^{2^{x-1}-1} \tfrac{(x-1)P(x)}{l_2}\right). \end{aligned} \quad (22)$$

C. Integration of EGEC decoder into a receiver

At the receiver of Figure 1, QPSK demodulation, demultiplexing, depuncturing as well as deinterleaving π_2^{-1} and π_3^{-1}, URC decoding as well as deinterleaving π_1^{-1} and π_4^{-1} is performed before invoking the EGEC(UEC) and EGEC(FLC-CC) decoders. Note that the receiver is required to employ the same pseudo-random interleaver designs as the transmitter. However, the entire set of interleavers can be generated independently by both the transmitter and receiver using only a single pseudo-random number generator seed. This seed may be hard-coded into both the transmitter or receiver, or may be reliably conveyed using only a very small amount of side information. As shown in Figure 1, it is necessary to operate the EGEC(UEC) decoder before the EGEC(FLC-CC) decoder, since the former outputs the sub-symbol vector $\hat{\mathbf{x}}$, which is required as an input to the latter. The extrinsic LLR vector $\tilde{\mathbf{z}}^{\mathrm{e}}$ of Figure 1 may be iteratively exchanged with the serially concatenated URC decoder. In-turn, the URC decoder may also iteratively exchange extrinsic LLRs with the demodulator [24], in order to avoid capacity loss, when a mapping scheme other than Gray coding or a higher-order modulation scheme is employed. Since the combination of the URC decoder and the demodulator will also have an EXIT curve that reaches the $(1, 1)$ point at the top right corner of the EXIT chart [25], iterative decoder convergence towards an approximation of the Maximum Likelihood (ML) performance is facilitated [26]. After completing the EGEC(UEC) decoding process, a similar iterative operation is performed for the EGEC(FLC-CC) decoder.

Following the completion of both the EGEC(UEC) and the EGEC(FLC-CC) iterative decoding processes, the sub-symbol vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{t}}$ are input to the sub-symbol recombination block S^{-1} of Figure 1, which recomposes the symbol vector $\hat{\mathbf{d}}$, according to (8). Observe that since the sub-symbol vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{t}}$ are guaranteed to comprise a number of symbols, the symbol vector $\hat{\mathbf{d}}$ will also comprise a number of symbols.

IV. Near-Capacity Performance of EGEC Codes and UEP Design

Near-capacity operation is achieved, when reliable communication can be maintained at transmission throughputs η that approach the Discrete-input Continuous-output Memoryless Channel (DCMC) capacity C that is associated with $M = 4$ QPSK modulation and uncorrelated narrowband Rayleigh fading. This is facilitated, if the following conditions are satisfied [20]:

1) The area A_1^{o} beneath the inverted EGEC(UEC) EXIT curve is required to approach the corresponding coding rate R_1^{o};
2) Likewise, the area A_2^{o} beneath the inverted EGEC(FLC-CC) EXIT curve is required to approach the corresponding coding rate R_2^{o};

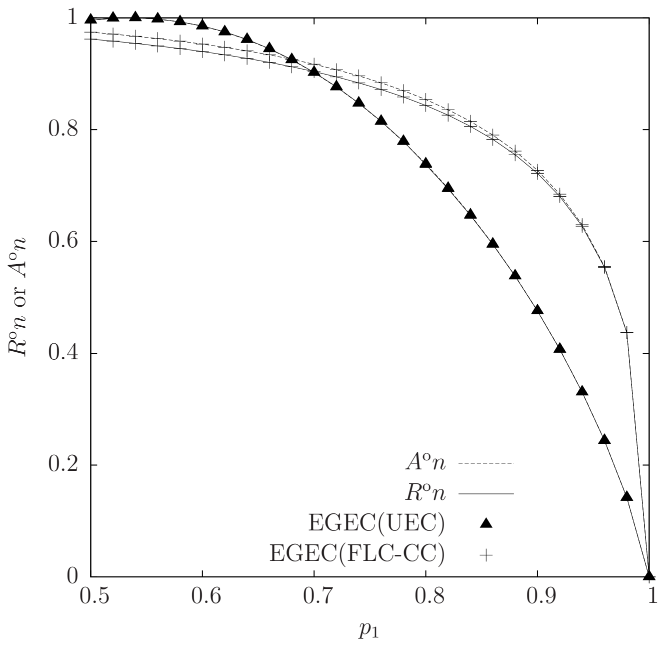


Fig. 4. Plots of $R^{\mathrm{o}}n$ and $A^{\mathrm{o}}n$ that are obtained for the EGEC scheme, in the case where the system input symbol values of $\mathbf{d}$ obey a zeta distribution having the parameter p_1. Here, R^{o} is the coding rate, A^{o} is the area beneath the inverted EXIT curve and n is the codeword length of the corresponding scheme. The value of $A^{\mathrm{o}}n$ is provided for an EGEC(UEC) code having $r_1 = 4$ states, as well as for an EGEC(FLC-CC) code having $x_{\max} = 3$.

3) The URC EXIT curves are required to satisfy $A_1^{\mathrm{i}} = C/[R_1^{\mathrm{i}} \log_2(M)]$ and $A_2^{\mathrm{i}} = C/[R_2^{\mathrm{i}} \log_2(M)]$.

If these three conditions are satisfied, then near-capacity operation will be achieved when the shape of URC decoders' EXIT curves are matched to those of the EGEC(UEC) and EGEC(FLC-CC) decoders. This creates narrow, but marginally open EXIT chart tunnels, which facilitate iterative decoding convergence towards an approximation of the ML performance.

When the RVs in the vector $\mathbf{X}$ obey the zeta distribution of (9), Figure 4 suggests that the first two of the above-mentioned conditions are satisfied. This figure plots the EGEC(UEC) coding rate R_1^{o} of (14) and the EGEC(FLC-CC) coding rate R_2^{o} of (19), when multiplied with the codeword lengths n_1 and n_2, respectively. Note that since the proposed EGEC code is a universal code, its coding rates are non-zero, regardless of the parameter value $0 < p_1 < 1$ employed for the zeta distribution of (1). Furthermore, Figure 4 plots the product of n_1 and the EGEC(UEC) EXIT area A_1^{o} of (21), for the case where the trellis decoder employs $r_1 = 4$ states. Likewise, the product of n_2 and the EGEC(FLC-CC) EXIT area A_2^{o} of (22) is plotted in Figure 4, for the case where $x_{\max} = 3$ is employed. Figure 4 shows that in all cases, the EXIT chart areas A_1^{o} and A_2^{o} approach the corresponding coding rates R_1^{o} and R_2^{o}. Figure 5 plots the discrepancy between $A_1^{\mathrm{o}}n_1$ and $R_1^{\mathrm{o}}n_1$ as a function of the number of EGEC(UEC) states r_1, where the source symbols of $\mathbf{d}$ obey the zeta distribution of (1) for various values for the parameter p_1. Note that in all the scenarios considered, the discrepancy becomes less than 10^{-3} for $r_1 = 4$, demonstrating that the EGEC(UEC) code imposes only an insignificant amount of capacity loss. Therefore, $r_1 = 4$ represents an attractive trade-off between facilitating near-capacity operation and maintaining a low trellis complexity. Similarly, the discrepancy between $A_2^{\mathrm{o}}n_2$ and $R_2^{\mathrm{o}}n_2$ is plotted for the EGEC(FLC-CC) code as a function of $x_{\max}$ in Figure 6, for zeta distributions having various values for the parameter p_1. Note that in all the scenarios considered, the discrepancy is around 10^{-2} for $x_{\max} = 3$, demonstrating that the EGEC(FLC-CC) code imposes only an insignificant amount of capacity loss. Therefore, $x_{\max} = 3$ represents an attractive trade-off between facilitating near-capacity operation

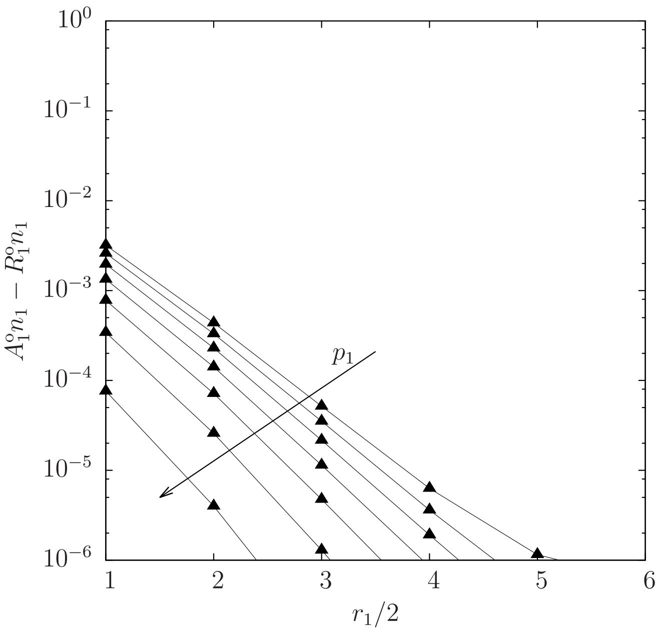


Fig. 5. The discrepancy between $A_1^{\mathrm{o}}n_1$ and $R_1^{\mathrm{o}}n_1$ that results when EGEC(UEC) codes having various numbers of states r_1 are employed to encode symbol values having zeta distributions with the parameters $p_1 \in \{0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$.

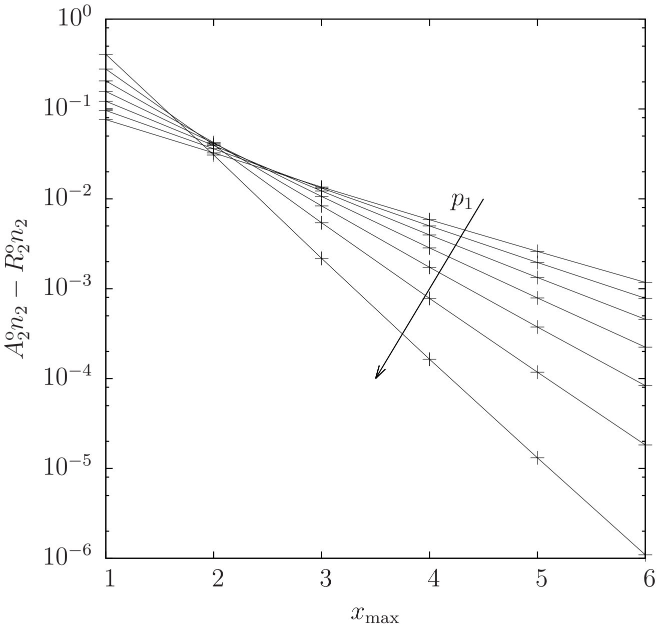


Fig. 6. The discrepancy between $A_2^{\mathrm{o}}n_2$ and $R_2^{\mathrm{o}}n_2$ that results when EGEC(FLC-CC) codes having various values for the parameter $x_{\max}$ are employed to encode symbol values having zeta distributions with the parameters $p_1 \in \{0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$.

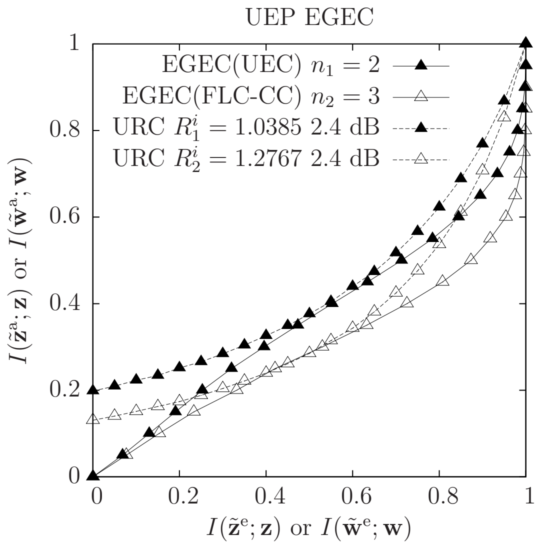


Fig. 7. EXIT charts of the UEP EGEC scheme. Here, the symbols of $\mathbf{d}$ obey a zeta distribution having $p_1 = 0.7967$ and the codewords comprise the numbers of bits n_1 and n_2. Furthermore, the punctured URC decoders adopt the coding rates $R_1^{\rm i}$ and $R_1^{\rm i}$, for Gray-coded QPSK modulation onto an uncorrelated narrowband Rayleigh fading channel having various $E_{\rm b}/N_0$. The EXIT curves are provided for an EGEC(UEC) code having $r_1 = 4$ states, as well as for an EGEC(FLC-CC) code having $x_{\max} = 3$.

and maintaining a low computational complexity. Note that unlike the EGEC(UEC), the EGEC(FLC-CC) EXIT chart area $A_2^{\rm o}$ is independent of the number of states r_2 employed in the CC. Finally, the third condition is satisfied by a punctured URC code, as discussed in [20].

Figure 4 shows that the areas beneath the EGEC(UEC) and EGEC(FLC-CC) EXIT curves $A_1^{\rm o}$ and $A_2^{\rm o}$ are different from each other in general. Owing to this, different areas are required beneath the URC EXIT curves $A_1^{\rm i}$ and $A_2^{\rm i}$, so that narrow but still open EXIT chart tunnels can be created simultaneously for both the EGEC(UEC) and EGEC(FLC-CC) codes. This can be achieved by employing different values for the coding rates $R_1^{\rm i}$ and $R_2^{\rm i}$, hence exhibiting UEP. More specifically, when decreasing one of these coding rates, the other should be increased, in order to maintain the same throughput η and facilitate a fair comparison with the Equal Error Protection (EEP) scheme in which $R_1^{\rm i} = R_2^{\rm i}$.

The advantages of UEP may be illustrated by comparing Figures 3 and 7, which consider the scenario where the symbols of $\mathbf{d}$ obey a zeta distribution having $p_1 = 0.7967$. More specifically, Figure 3 considers an EEP scheme having an $n_1 = 2$-bit EGEC(UEC) code and an $n_2 = 2$-bit EGEC(FLC-CC) code, as well as inner coding rates of $R_1^{\rm i} = 1$ and $R_2^{\rm i} = 1$, which gives a throughput of $\eta = 0.7620$ bit/s/Hz. Observe in Figure 3 that an open EXIT chart tunnel is created by the EGEC(UEC) code at an $E_{\rm b}/N_0$ of 1.9 dB, but by contrast this is not facilitated until reaching an $E_{\rm b}/N_0$ of 2.9 dB for the EGEC(FLC-CC) code. Let us now consider an UEP scheme, which employs an $n_1 = 2$-bit EGEC(UEC) code and an $n_2 = 3$-bit EGEC(FLC-CC) code, as well as inner coding rates of $R_1^{\rm i} = 1.0385$ and $R_2^{\rm i} = 1.2767$, in order to achieve the same throughput of $\eta = 0.7620$ bit/s/Hz. Observe in Figure 7 that this scheme can simultaneously create open EXIT chart tunnels for both the EGEC(UEC) code and the EGEC(FLC-CC) code at an $E_{\rm b}/N_0$ of 2.4 dB, offering 0.5 dB of gain over the EEP scheme.

In general, an UEP scheme can be designed by appropriately choosing n_1, n_2, R_1 and R_2 for ensuring that the desired throughput η is achieved and the two EXIT chart tunnels become marginally open at the same $E_{\rm b}/N_0$ value. Here, we recommend the value of $n_1 = n_2 = 2$ for the codeword lengths when possible, while the value of $n_2 = 3$ whenever necessary to achieve the desired throughput η.

Table II provides parametrizations for both EEP and UEP EGEC schemes, designed for transmitting symbols that obey the zeta distribution of (1). We parametrize the zeta distribution using $p_1 \in \{0.9000, 0.7967, 0.6940, 0.6000\}$, which represents a wide selection of the p_1 values shown in Figure 4. Here, $p_1 = 0.7967$ is chosen for consistency with [12], while $p_1 = 0.6940$ is chosen because it is the specific value where $R_1^{\rm o}$ and $R_2^{\rm o}$ are equal to each other. The EGEC(UEC) scheme adopts the $n_1 = 2$-bit $r_1 = 4$-state UEC trellis of [12, Figure 3(b)], while the EGEC(FLC-CC) scheme adopts $x_{\max} = 3$, as well as the $r_2 = 4$-state recursive CC of [12, Table II] having either $n_2 = 2$-bit codewords or $n_2 = 3$-bit codewords, as appropriate. As discussed above, we select $r_1 = 4$ UEC states, since this is sufficiently high for imposing only an insignificant amount of capacity loss, while $r_2 = 4$ CC states were selected, because having a higher number of states was found to be detrimental in [12].

Table II also characterizes the complexity of the EEP and UEP EGEC schemes. Here, the complexity is quantified by the average number of Add, Compare and Select (ACS) operations performed per decoding iteration and per symbol in the vector $\mathbf{d}$. This is justified, since the EGEC(UEC) trellis decoder, the EGEC(FLC-CC) decoder and the URC decoder operate entirely on the basis of addition, subtraction and $\max^*$ operations, while all other components in Figure 1 may be considered to have a relatively insignificant complexity [16], [18]. As in [16], we assume that each $\max^*$ operation may be completed using five ACS operations, while the addition and subtraction operations each require a single ACS operation. As shown in Table II, the complexity tends to increase as the zeta distribution parameter p_1 is reduced, which is due to the resultant increases in the average codeword lengths l_1 and l_2. Furthermore, since the UEC EGEC schemes employ $n_2 = 3$-bit EGEC(FLC-CC) codes, they are associated with a higher complexity than the corresponding EEP schemes, which employ shorter $n_2 = 2$-bit codes.

Table II provides the $E_{\rm b}/N_0$ values, where the DCMC capacity C becomes equal to the throughput η of each scheme considered. These $E_{\rm b}/N_0$ values represent *capacity bounds*, above which it is theoretically possible to achieve reliable communication, provided that the scheme facilitates near-capacity operation. Furthermore, the specific $E_{\rm b}/N_0$ values, where we have $A^{\rm i} = A^{\rm o}$ are provided for each scheme considered in Table II. These *area bounds* represent the lowest $E_{\rm b}/N_0$ values, where it is theoretically possible to create an open EXIT chart tunnel, provided that the EGEC and URC EXIT function have shapes that match each other. Note that the discrepancy between the capacity bound and the area bound of each scheme represents *capacity loss*. For each of the UEP schemes considered, the capacity loss is less

TABLE II
OUTER CODING RATE R^{o}, INNER CODING RATE R^{i} AND THROUGHPUT η FOR THREE SCHEMES WITH DIFFERENT p_1 VALUES.

p_1	Scheme			n	r	R^{o}	A^{o}	R^{i}	η	E_{b}/N_0 [dB] for $C = \eta$	E_{b}/N_0 [dB] for $A^{\mathrm{i}} = A^{\mathrm{o}}$	E_{b}/N_0 [dB] for open tunnel	Complexity
0.9	EGEC	EEP	UEC	2	4	0.2378	0.2378	1.0578	0.5272	0.01	2.4	3.9	267
			FLC-CC	2	4	0.3609	0.3636						
		UEP	UEC	2	4	0.2378	0.2378	1.1251			0.1	1.0	286
			FLC-CC	3	4	0.2406	0.2424	1					
	UEC			2	4	0.2636	0.2682	1			0.1	1.5	250
	EG-CC			2	4	0.2492	0.3247	1.0578			1.6	2.4	257
0.7967	EGEC	EEP	UEC	2	4	0.3721	0.3721	1	0.7620	0.84	1.6	2.9	338
			FLC-CC	2	4	0.4229	0.4283						
		UEP	UEC	2	4	0.3721	0.3721	1.0385			0.9	2.4	379
			FLC-CC	3	4	0.2820	0.2855	1.2767					
	UEC			2	4	0.3810	0.4041	1			1.3	2.5	331
	EG-CC			2	4	0.3810	0.4410	1			2.0	3.0	322
0.6940	EGEC	EEP	UEC	2	4	0.4533	0.4535	1	0.9066	1.43	1.5	2.5	431
			FLC-CC	2	4	0.4533	0.4599						
	UEC			2	4	0.3112	0.3654	1.4565			2.7	4.5	614
	EG-CC			2	4	0.4533	0.4877	1			2.0	3.0	410
0.6	EGEC	EEP	UEC	2	4	0.4906	0.4910	1	0.9690	1.69	1.8	2.8	547
			FLC-CC	2	4	0.4699	0.4766						
	EG-CC			2	4	0.4845	0.4998	1			2.0	3.0	522

than 0.1 dB, demonstrating that the proposed EGEC scheme facilitates near-capacity operation. Finally, Table II provides the *tunnel bound* of each scheme, which quantifies the lowest E_{b}/N_0 value, where an open EXIT chart tunnel can be created upon employing a two-state accumulator for the URC code. Note that in all cases, our experiments revealed that two-state URC codes facilitate the creation of open tunnels at lower E_{b}/N_0 values than four- or eight-state URCs, as well as having a lower decoding complexity. The proposed EGEC schemes facilitate reliable communication at E_{b}/N_0 values that exceed the corresponding tunnel bound, provided that the symbol vector $\mathbf{d}$ comprises a sufficiently high number a of symbols.

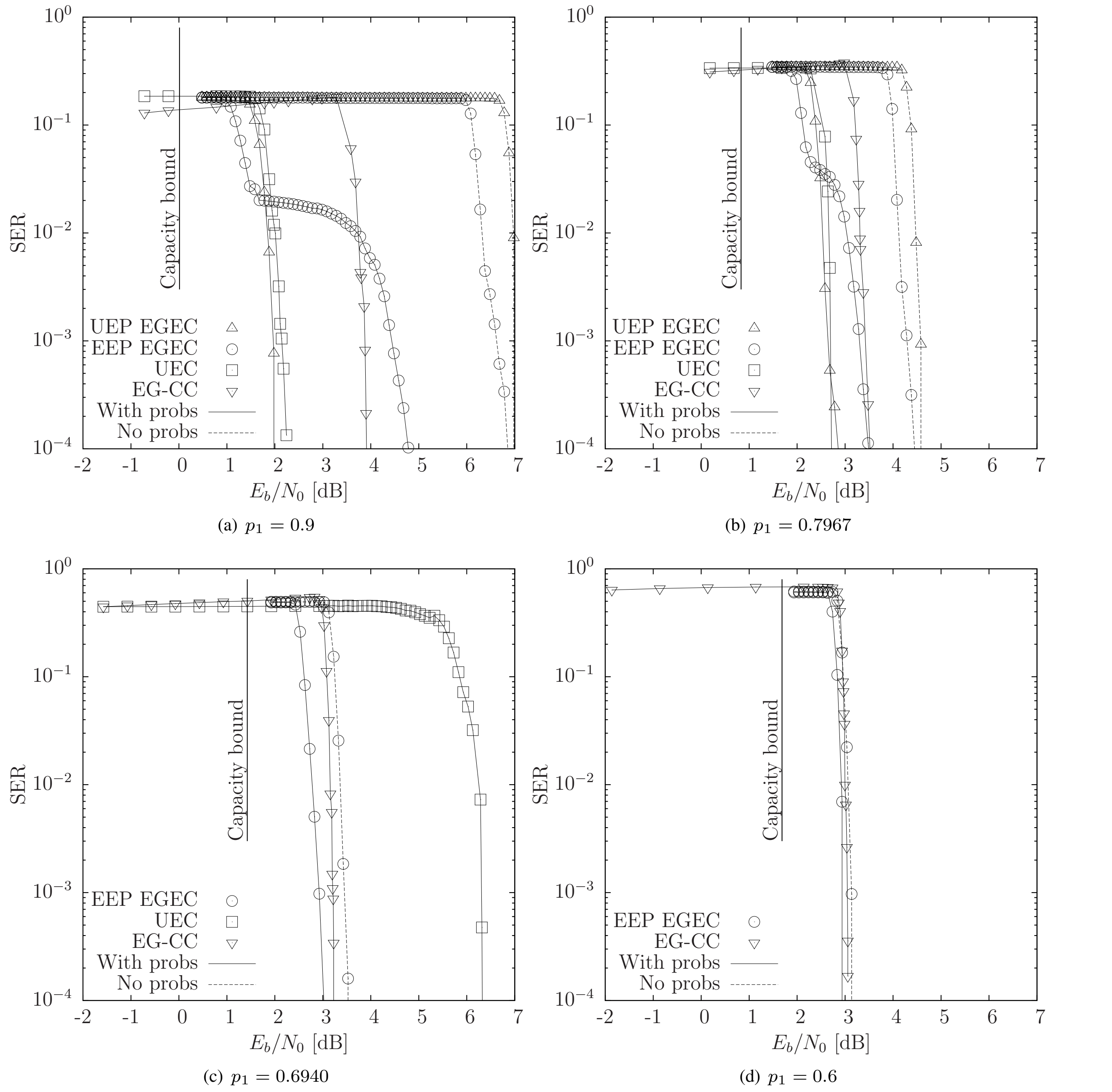
Note that the capacity loss analysis of this section relies upon (21) and (22), which are specific to the scenario where the LLRs of $\tilde{\mathbf{z}}^{\mathbf{a}}$, $\tilde{\mathbf{u}}^{\mathrm{a}}$, $\tilde{\mathbf{v}}^{\mathrm{a}}$ and $\tilde{\mathbf{w}}^{\mathrm{a}}$ adopt distributions that may be modeled by transmission over a BEC. However, the authors of [20] showed that the area beneath an EXIT curve is not significantly affected by the LLR distribution. Indeed, the results of Section V demonstrate that the proposed EGEC code is also capacity-approaching when the LLR distribution corresponds to communication over an uncorrelated narrowband Rayleigh fading channel.

V. PERFORMANCE COMPARISON WITH THE BENCHMARKERS

In this section, we compare the proposed EEP and UEP EGEC schemes to the UEC and EG-CC benchmarkers of [12]. Table II provides parametrizations for these benchmarkers, which offer the same throughput η as our EGEC schemes. Here, the UEC benchmarker adopts the $n = 2$-bit $r = 4$-state UEC trellis of [12, Figure 3(b)], since this is recommended in [16]. Furthermore, the UEC is serially-concatenated with a URC, for the sake of facilitating iterative decoding. While the proposed EGEC schemes and the UEC benchmarker constitute examples of JSCCs, the EG-CC benchmarker represents SSCC. More specifically, the EG-CC benchmarker employs an EG code for source coding, while an iteratively-decoded serial-concatenation of a CC and a URC is employed for separate channel coding. Here, we select the $n = 2$-bit $r = 4$-state CC of [12, Table II], since higher numbers of states were found to be detrimental in [12]. As in the proposed EGEC schemes, the UEC and EG-CC benchmarkers employ two-state accumulators for their URCs, since these were found to yield open EXIT chart tunnels at the lowest E_{b}/N_0 values. Note that the UEC and EG-CC benchmarkers offer fair and natural comparisons with the proposed EGEC schemes, since they all employ simple unary, FLC or EG codewords, as well as UEC or CC trellises having four states. Furthermore, EG codes and CCs are employed in numerous multimedia transmission standards, such as H.264 [11] and DVB-T [27].

Figure 8 characterizes the Symbol Error Ratio (SER) performance of the schemes parametrized in Table II. We consider the transmission of source symbol vectors $\mathbf{d}$ comprising $a = 2 \cdot 10^4$ symbols, which we found to be typical of the number of EG-encoded symbols that appear in a H.264 slice [11]. Therefore, the SER performance of Figure 8 may be considered to be achievable, without imposing any additional latency in multimedia applications. We employed QPSK modulation for transmission over an uncorrelated narrowband Rayleigh fading channel, since this is representative of transmission over realistic wireless channels and because this facilitates direct comparison with the results of [12], [16]. In the receivers, iterative decoding was continued until convergence was achieved.

As shown in Figure 8, the proposed EGEC schemes facilitate reliable communication within 1.2 dB of the capacity bound and consistently offer the best SER performance for each of the p_1 values considered. This consistency is a key benefit of the proposed EGEC scheme, because while it offers only a small gain over the best of the two benchmarkers in each case, the performance of these benchmarkers is particularly inconsistent. More explicitly, while the proposed EGEC scheme offers only a marginal gain over the UEC benchmarker for $p_1 \in \{0.9, 0.7967\}$, this gain becomes 3.4 dB for $p_1 = 0.6940$, owing to the severe puncturing that the UEC scheme requires in this case [16]. Furthermore, the UEC benchmarker cannot be invoked for $p_1 = 0.6$, since



(a) $p_1 = 0.9$

(b) $p_1 = 0.7967$

(c) $p_1 = 0.6940$

(d) $p_1 = 0.6$

Fig. 8. The SER performance that is obtained following the achievement of iterative decoding convergence in the EGEC scheme of Figure 1, as well as in the UEC and EG-CC benchmarkers of [12], when transmitting frames comprising $a = 2 \cdot 10^4$ symbols using QPSK modulation over an uncorrelated narrowband Rayleigh fading channel. The plots labeled 'With probs' and 'No probs' indicate the SER performance that is achievable when the source distribution $P(d)$ is known and unknown to the receiver, respectively.

the average unary codeword length $l_{\text{Unary}(d_i)}$ becomes infinite in this case. Similarly, while the proposed EGEC scheme offers only a marginal gain over the EG-CC benchmarker of $p_1 \in \{0.6940, 0.6\}$, this gain becomes 1.9 dB for $p_1 = 0.9$ and 0.8 dB for $p_1 = 0.7967$, as shown in Figure 8.

The complexity of the proposed EGEC schemes is compared to that of the benchmarkers in Table II. For each p_1 value considered, it can be seen that the complexity of the various schemes is similar, demonstrating that the gains offered by the proposed EGEC schemes are not accrued at the cost of a significantly increased complexity. Instead, these gains may be attributed to the EGEC schemes' avoidance of the capacity loss suffered by the EG-CC benchmarker, as well as due to avoiding the UEC scheme's requirement for excessive puncturing, when $p_1 = 0.6940$.

Note that the UEP EGEC schemes offer superior SER performance over the EEP schemes, as shown in Figure 8. The stair-case shaped SER performance of the EEP schemes may be attributed to the opening of the EGEC(UEC) EXIT chart tunnel at a lower E_b/N_0 value than the EGEC(FLC-CC) EXIT chart tunnel, as shown in Table II. At E_b/N_0 values where the EGEC(UEC) EXIT chart tunnel is open, but the EGEC(FLC-CC) EXIT chart tunnel is closed, symbols having a value of $d_i = 1$ are typically correctly decoded, since these symbols are conveyed without using any EGEC(FLC-CC)-encoded bits. By contrast, symbols having values of $d_i > 1$ are typically incorrectly decoded in this case, owing to transmission errors affecting their corresponding EGEC(FLC-CC)-encoded bits.

A similar phenomenon may be observed for the EG-CC benchmarker, increasing the SER as the E_b/N_0 value is increased towards the threshold value, where the EXIT chart tunnel becomes open. More specifically, at very low E_b/N_0 values, the information received over the channel is associated with a very low confidence and hence the iterative decoding process is dominated by the *a priori* knowledge that the symbol value $d_i = 1$ is most likely. This causes a value of 1 to be selected for all symbols in the vector $\hat{\mathbf{d}}$, resulting in an SER of $(1 - p_1)$. As the E_b/N_0 value is increased, the information received over the channel has a greater influence on the iterative decoding process, causing values other than unity to be increasingly selected for some symbols in $\hat{\mathbf{d}}$. However, these non-unity values are typically allocated to the wrong symbols, owing to the loss of synchronization that is caused by the frequent occurrence of decoding errors at E_b/N_0 values below the threshold. This effect occurs more frequently as the E_b/N_0 value is further increased towards the threshold value, causing the SER to increase, as shown in Figure 8. By contrast, for E_b/N_0 values above the threshold, the decoding errors are mitigated and the SER reduces rapidly.

Note that throughout the discussion above, it is assumed that the receiver of the proposed EGEC scheme has knowledge of the average unary codeword length l_1. Furthermore, we assume knowledge of the sub-symbol probabilities $P(x)$ for $x \leq r_1/2 - 1$, as well as the conditional sub-symbol probabilities $P(t|x)$ for all pairs of t and x where $x \leq x_{\max}$. These probabilities can be calculated with knowledge of $P(d)$ for $d \leq 2^{\max(x_{\max}, r_1/2-1)} - 1$. Since we employ $r_1 = 4$ and $x_{\max} = 3$, the SER results presented above may be obtained with knowledge of only the first seven symbol probabilities $P(d)$, as well as l_1. However, Figure 8 shows that when the channel SNR is sufficiently high, the proposed EGEC receiver facilitates a low SER, even if it does not have access to this information. This may be exploited to recover a sufficient number of symbol vectors $\hat{\mathbf{d}}$, in order to heuristically estimate the small amount of required information, facilitating the near-capacity communication of subsequent symbol vectors.

VI. Conclusions

In this paper we have proposed novel EGEC codes for the near-capacity transmission of symbol values that are randomly selected from a source set having an infinite cardinality. In contrast to the UEC code previously proposed for the same purpose, our EGEC code is a universal code, facilitating the transmission of symbol values that are randomly selected using any mototonic probability distribution. When the source symbols obey a particular zeta probability distribution, our EGEC scheme was shown to offer a 3.4 dB gain over a UEC benchmarker, when QPSK modulation is employed for transmission over an uncorrelated narrowband Rayleigh fading channel. In the case of another zeta probability distribution, our EGEC scheme was shown to offer a 1.9 dB gain over a SSCC benchmarker. Furthermore, we considered a wide range of zeta probability distributions and our EGEC scheme was found to offer gains over the relevant benchmarkers in each case.

References

[1] C. E. Shannon, *Mathematical Theory of Communication*. University of Illinois Press, 1963.

[2] B. Ryabko and J. Rissanen, "Fast adaptive arithmetic code for large alphabet sources with asymmetrical distributions," in *Proc. 2002 IEEE Int. Symp. Inform. Theory*, p. 319.

[3] J. Ziv and A. Lempel, "Compression of indivdual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.

[4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 457–458, Aug. 1996.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)," in *Proc. 1993 IEEE Int. Conf. on Communications*, vol. 2, pp. 1064–1070.

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

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

[8] S. Even and M. Rodeh, "Economical encoding of commas between strings," *Commun. ACM*, vol. 21, no. 4, pp. 315–317, 1978. Available: http://dblp.uni-trier.de/db/journals/cacm/cacm21.html#EvenR78;http://doi.acm.org/10.1145/359460.359480;http://www.bibsonomy.org/bibtex/255477367a17c860a3e8543febfb75cc7/dblp

[9] Q. F. Stout, "Improved prefix encodings of the natural numbers (corresp.)," *IEEE Trans. Inf. Theory*, vol. 26, no. 5, pp. 607–609, 1980. Available: http://dblp.uni-trier.de/db/journals/tit/tit26.html#Stout80;http://doi.ieeecomputersociety.org/10.1109/TIT.1980.1056237;http://www.bibsonomy.org/bibtex/2d76e21d35c99cb9963ecbcd237b3fc16/dblp

[10] A. S. Fraenkel and S. T. Klein, "Robust universal complete codes for transmission and compression," *Discrete Applied Mathematics*, vol. 64, no. 1, pp. 31–55, 1996. Available: http://dblp.uni-trier.de/db/journals/dam/dam64.html#FraenkelK96;http://dx.doi.org/10.1016/0166-218X(93)00116-H;http://www.bibsonomy.org/bibtex/2ca088a7d65bf1449bd409f80cb4b7a51/dblp

[11] Advanced video coding for generic audiovisual services, ITU-T Std. H.264, Mar. 2005.

[12] 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.*, 2013 (in press). Available: http://eprints.soton.ac.uk/341736/.

[13] N. L. Johnson, A. W. Kemp, and S. Kotz, *Univariate Discrete Distributions*. John Wiley & Sons, 2005.

[14] R. G. Maunder and L. Hanzo, "Genetic algorithm aided design of component codes for irregular variable length coding," *IEEE Trans. Commun.*, vol. 57, no. 5, pp. 1290–1297, May 2009. Available: http://eprints.ecs.soton.ac.uk/14470

[15] R. Gallager and D. van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. Inf. Theory*, vol. 21, no. 2, pp. 228–230, Mar. 1975.

[16] W. Zhang, R. G. Maunder, and L. Hanzo, "On the complexity of unary error correction codes for the near-capacity transmission of symbol values from an infinite set," in *Proc. 2013 IEEE Wireless Communications and Networking Conference*. Available: http://eprints.soton.ac.uk/344059/.

[17] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimising symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[18] M. Adrat, R. Vary, and J. Spittka, "Iterative source-channel decoder using extrinsic information from softbit-source decoding," in *Proc. 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2653–2656.

[19] D. Divsalar, S. Dolinar, and F. Pollara, "Serial concatenated trellis coded modulation with rate-1 inner code," in *Proc. 2000 IEEE Global Telecommunications Conference*, vol. 2, pp. 777–782.

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

[21] J. Kliewer, A. Huebner, and D. J. Costello, "On the achievable extrinsic information of inner decoders in serial concatenation," in *Proc. 2006 IEEE Int. Symp. Inform. Theory*, pp. 2680–2684.

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

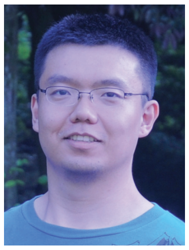
[23] J. Kliewer, N. Görtz, and A. Mertins, "Iterative source-channel decoding with Markov random field source models," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3688–3701, Oct. 2006.

[24] M. Tuchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *Proc. 2002 IEEE Global Telecommunications Conference*, vol. 2, pp. 1358–1362.

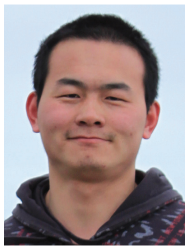
[25] R. Maunder and L. Hanzo, "Iterative decoding convergence and termination of serially concatenated codes," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 216–224, Jan. 2010.

[26] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. 1998 Allerton Conf. on Communications, Control and Computing*, pp. 201–210.

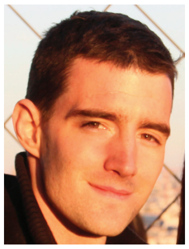
[27] Digital Video Broadcasting (DVB): Framing structure, channel coding and modulation for digital terrestrial television, ETSI Std. EN 300 744.



Tao Wang received the B.S. degree in information engineering from the University of Science and Technology of Beijing (USTB), Beijing, China, in 2006. He received M.Sc. degree in communication from University of Southampton, Southampton, U.K in 2008. He is currently working toward the Ph.D. degree with the Communications Research Group, Electronics and Computer Science, University of Southampton, Southampton, UK. His current research interests include joint source/channel coding and distributed video coding.



Wenbo Zhang received the M.E. degree in Information and Communication Engineering from the University of Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2011. He is currently working toward the Ph.D. degree with the Communications Research Group, Electronics and Computer Science, University of Southampton, Southampton, UK. His current research interests include joint source/channel coding and variable length coding.



Robert G. Maunder (http://users.ecs.soton.ac.uk/rm) has studied with Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honors BEng in Electronic Engineering in July 2003, as well as a PhD in Wireless Communications and a lectureship in December 2007. Rob's research interests include joint source/channel coding, iterative decoding, irregular coding and modulation techniques. He has published a number of IEEE papers in these areas.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded the honorary doctorate "Doctor Honoris Causa" by the Technical University of Budapest. During his 35-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised 80 Ph.D. students, co-authored 20 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, coauthored 1300+ research entries at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing a 100-strong academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European IST Programme and the Mobile Virtual Centre of Excellence (VCE), UK. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE VTS. During 2008 - 2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing. His research is funded by the European Research Council's Senior Research Fellow Grant. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk