Hybrid Concatenated Codes and Iterative Decoding

D. Divsalar and F. Pollara
Communications Systems and Research Section

A hybrid concatenated code with two interleavers is the parallel concatenation of an encoder, which accepts the permuted version of the information sequence as its input, with a serially concatenated code, which accepts the unpermuted information sequence. The serially concatenated code consists of an outer encoder, an interleaver, and an inner encoder. An upper bound to the average maximum-likelihood bit-error probability of the hybrid concatenated convolutional coding schemes is obtained. Design rules for the parallel, outer, and inner codes that maximize the interleaver's gain and the asymptotic slope of the error-probability curves are presented. Finally, a low-complexity iterative decoding algorithm that yields performance close to maximum-likelihood decoding is proposed. A special case of hybrid concatenated code where the outer code is a repetition code is analyzed, and another special case called self-concatenated code is introduced. Comparisons with parallel concatenated convolutional codes, known as "turbo codes," and with recent serially concatenated convolutional codes are discussed, showing that the new scheme offers better performance at very low bit-error rates when low-complexity codes are used. An example of the proposed scheme for deep-space communications is presented.

I. Introduction

Concatenated coding schemes have been studied by Forney in [1]. They consist of the cascade of an *inner* code and an *outer* code, which, in Forney's approach, would be a relatively short inner block code, or a convolutional code with maximum-likelihood Viterbi decoding, and a long high-rate nonbinary Reed–Solomon outer code decoded by an algebraic error-correction algorithm. Concatenated codes have since then evolved as a standard for those applications where very high coding gains are needed, such as deep-space applications. Alternative solutions for concatenation have also been studied in [2], [3], and [19]. In [19], Tanner proposed a method for construction of long error-correcting codes from shorter ones based on bipartite graphs and an iterative decoding.

Turbo codes [4] are *parallel* concatenated convolutional codes (PCCCs) using two constituent codes. Parallel concatenation was extended to more than two codes in [5] (see also [21]). These codes were analyzed in [6,22–24].

Using the same ingredients, namely convolutional encoders and interleavers, *serially* concatenated convolutional codes (SCCCs) have been shown to yield performances comparable and, in some cases, superior to turbo codes [14]. A third choice is a *hybrid* concatenation of convolutional codes (HCCC). In this article, we consider, as an example of hybrid concatenated code, only the parallel concatenation

of a convolutional code with a serially concatenated convolutional code. Serial concatenation of an outer convolutional code with an inner turbo code and other types of hybrid concatenated codes are considered in [15].

These concatenated coding schemes use a suboptimum decoding process based on iterating an a posteriori probability (APP) algorithm [7] applied to each constituent code. A soft-input, soft-output (SISO) APP module described in [13] was used. As an example, we will show the results obtained by decoding a hybrid concatenation of three codes with very high coding gain for deep-space applications.

For HCCCs, we obtain analytical upper bounds to the performance of a maximum-likelihood (ML) decoder using analytical tools and notations introduced in [6] and [9]. We propose design rules leading to the optimal choice of constituent convolutional codes that maximize the *interleaver gain* [6] and the asymptotic code performance, and we present a new iterative decoding algorithm with limited complexity using the SISO module. Comparisons with turbo codes and serially concatenated codes of the same complexity and decoding delay are discussed.

In Section II, we derive analytical upper bounds to the bit-error probability of HCCCs using the concept of "uniform interleavers" that decouples the output of the outer encoder from the input of the inner encoder, and the input of the parallel code from the input of the outer code. In Section III, we propose design rules for HCCCs through an asymptotic approximation of the bit-error probability bound, assuming long interleavers or large signal-to-noise ratios (SNRs). Section IV describes a new iterative decoding algorithm. Section V considers a special case of HCCC when the outer code is a repetition code. Furthermore, if the parallel code is a one-state, rate 1 code (no code), then we obtain an encoding structure that we call "self-concatenated" since there is only one convolutional code involved. The self-concatenated code is analyzed, a simple iterative decoding structure is proposed, and simulation results are shown in Section VI. Simulation results for an example of HCCC for deep-space communications are presented in Section VII.

II. Analytical Bounds on the Performance of Hybrid Concatenated Codes

Consider a linear (n, k) block code C with code rate $R_c = k/n$ and minimum distance h_m. An upper bound on the bit-error probability (using the union bound) of the block code C over additive white Gaussian noise (AWGN) channels, with coherent detection and using maximum-likelihood decoding, can be obtained as

$$P_b \leq \sum_{h=h_m}^{n} \sum_{w=1}^{k} \frac{w}{k} A_{w,h}^{C} Q\left(\sqrt{2R_c\, h \frac{E_b}{N_0}}\right) \tag{1}$$

where E_b/N_0 is the signal-to-noise ratio per bit and $A_{w,h}^{C}$ represents the number of codewords of the block code C having output weight h and associated with input sequences of weight w. The $A_{w,h}^{C}$ is the input–output weight coefficient (IOWC). The function $Q(\sqrt{2R_c\ h\ E_b/N_0})$ represents the pairwise error probability, which is a monotonic decreasing function of the signal-to-noise ratio and the output weight, h. The Q function is defined as $Q(x) = (1/\sqrt{2\pi}) \int_x^\infty e^{-(t^2/2)} dt$.

If we construct an equivalent block code from the convolutional code, this bound applies to convolutional codes as well. Obviously, this result also applies to concatenated codes, including parallel and serial concatenations, as well as to the hybrid concatenated codes discussed in this article. As soon as we obtain the input–output weight coefficients $A_{w,h}^{C}$ for an HCCC code, we can compute its performance.

A. Hybrid Concatenated Convolutional Codes

The structure of a hybrid concatenated convolutional code is shown in Fig. 1. It is composed of three concatenated codes: the *parallel* code C_p with rate $R_c^p = k_p/n_p$ and equivalent block code representation $(N_1/R_c^p, N_1)$, the *outer* code C_o with rate $R_c^o = k_o/p_o$ and equivalent block code representation[1] $(N_1/R_c^o, N_1)$, and the *inner* code C_i with rate $R_c^i = p_i/n_i$ and equivalent block code representation $(N_2/R_c^i, N_2)$, with two interleavers N_1 and N_2 bits long, generating an HCCC C_H with overall rate R_c. For simplicity, we assume $k_p = k_o$ and $p_o = p_i \triangleq p$; then $R_c = k_o/(n_p + n_i)$.

Since the HCCC has two outputs, the upper bound on the bit-error probability in Expression (1) can be modified to

$$P_b \leq \sum_{h=h_m^p}^{n_1} \sum_{h=h_m^i}^{n_2} \sum_{w=w_m}^{k} \frac{w}{k} A_{w,h_1,h_2}^{C_H} Q\left(\sqrt{2R_c\,(h_1+h_2)\,\frac{E_b}{N_0}}\right) \tag{2}$$

where $A_{w,h_1,h_2}^{C_H}$ for the HCCC code C_H represents the number of codewords of the equivalent block code with output weight h_1 for the parallel code and output weight h_2 for the inner code associated with an input sequence of weight w; $A_{w,h_1,h_2}^{C_H}$ is the IOWC for the HCCC; w_m is the minimum weight of an input sequence generating the error events of the parallel code and the outer code; h_m^p is the minimum weight of the codewords of C_p; and h_m^i is the minimum weight of the codewords of C_i.

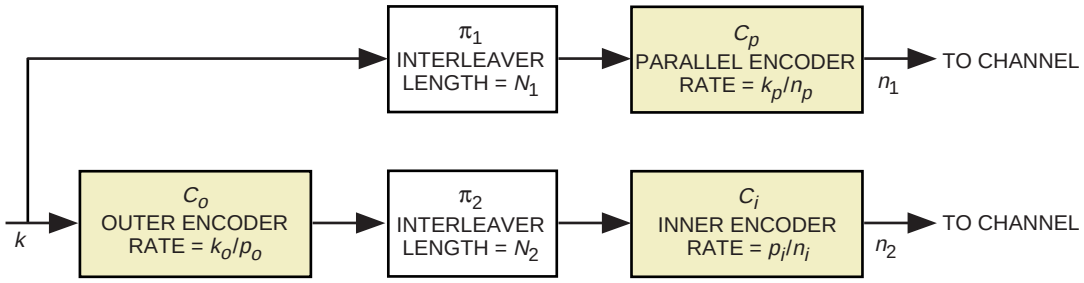

Fig. 1. A hybrid concatenated code.

B. Computation of $A_{w,h1,h2}^{CH}$ for Hybrid Concatenated Codes With Random Interleavers

If the input block k is large, then the computation of $A_{w,h_1,h_2}^{C_H}$ for fixed interleavers is an almost impossible task, except for the first few input and output weights. However, the *average* input–output weight coefficients, $A_{w,h_1,h_2}^{C_H}$, for hybrid concatenated codes with two interleavers can be obtained by averaging Expression (2) over all possible interleavers. This average is obtained by replacing the actual interleavers with abstract interleavers called *uniform interleavers* [6], defined as probabilistic devices that map a given input word of weight w into all its distinct $\binom{N}{w}$ permutations with equal probability $p = 1/\binom{N}{w}$, so that the input and output weights are preserved and where N represents the size of the interleaver.

With knowledge of the IOWC $A_{w,h_1}^{C_p}$ for the constituent parallel code, the IOWC $A_{w,l}^{C_o}$ for the constituent outer code, and the IOWC $A_{l,h_2}^{C_i}$ for the constituent inner code, using the concept of the uniform interleaver, the $A_{w,h_1,h_2}^{C_H}$ for the hybrid concatenated code can be obtained.

[1] Here the outer code is assumed to be a convolutional code, but conventional block codes, such as Hamming and Bose–Chaudhuri–Hocquenghem (BCH) codes, can be used as outer codes. To reduce the trellis complexity, short block codes can be used, say, m times, where m is the ratio of the input block size k of the HCCC over the input block size of the short block code.

According to the properties of uniform interleavers, the first interleaver transforms input data of weight w at the input of the outer encoder into all its distinct $\binom{N_1}{w}$ permutations at the input of the parallel encoder. Similarly, the second interleaver transforms a codeword of weight l at the output of the outer encoder into all its distinct $\binom{N_2}{l}$ permutations at the input of the inner encoder. As a consequence, each input data block weight w, through the action of the first uniform interleaver, enters the parallel encoder generating $\binom{N_1}{w}$ codewords of the parallel code C_p, and each codeword of the outer code C_o of weight l, through the action of the second uniform interleaver, enters the inner encoder generating $\binom{N_2}{l}$ codewords of the inner code C_i. Thus, the expression for the IOWC of the HCCC is

$$A^{C_H}_{w,h_1,h_2} = \sum_{l=0}^{N_2} \frac{A^{C_p}_{w,h_1} \times A^{C_o}_{w,l} \times A^{C_i}_{l,h_2}}{\binom{N_1}{w}\binom{N_2}{l}} \tag{3}$$

where $A^{C_o}_{w,l}$ is the number of codewords of the outer code of weight l associated with an input word of weight w.

Since we compute the average performance, this means that there will always be, for each value of the signal-to-noise ratio, at least a set of two particular interleavers yielding performance better than or equal to that of the two uniform interleavers. Using Eq. (3) in Expression (2), we can rewrite the upper bound in Expression (2) as

$$P_b(e) \le \sum_{h_1=h^p_m}^{N_1/R^p_c} \sum_{h_2=h^i_m}^{N_2/R^i_c} \sum_{w=w_m}^{N_1} \sum_{l=0}^{N_2} \frac{A^{C_p}_{w,h_1} \times A^{C_o}_{w,l} \times A^{C_i}_{l,h_2}}{\binom{N_1}{w}\binom{N_2}{l}} \frac{w}{N_1} Q\left(\sqrt{2R_c\,(h_1+h_2)\,\frac{E_b}{N_0}}\right) \tag{4}$$

Example 1. Consider a rate 1/4 HCCC formed by a parallel four-state recursive systematic convolutional code with rate 1/2, where the systematic bits of the parallel encoder (as for turbo codes) are not transmitted; an outer four-state nonrecursive convolutional code with rate 1/2; and an inner four-state recursive systematic convolutional code with rate 2/3, joined by two uniform interleavers of length $N_1 = N$ and $N_2 = 2N$, where N=20, 40, 100, 200, and 300.

The code generator matrices are shown in Table 1. Using Expression (4), we have obtained the bit-error probability curves shown in Fig. 2. The input–output weight coefficients of constituent codes were computed with an efficient recursive algorithm that we have developed. The performance shows a very significant interleaver gain, i.e., lower values of the bit-error probability for higher values of N.

III. Design of Hybrid Concatenated Codes

In the following, we use analytical tools, definitions, and notations introduced in [6] and [9]. The design of hybrid concatenated codes is based on the asymptotic behavior of the upper bound in Expression (4) for large interleavers. The reason for the good performances of parallel and serial concatenated codes with input block sizes of N symbols was that the normalized coefficients $A^C_{w,h}/N$ of a concatenated code decrease with interleaver size for all w and h. For a given signal-to-noise ratio and large interleavers, the maximum component of $A^{C_H}_{w,h_1,h_2}/N$ over all input weights w and output weights h_1 and h_2 is proportional to N^{α_M}, with corresponding minimum output weight $h(\alpha_M)$.[2] If $\alpha_M < 0$, then for a given SNR, the

[2] The α_M is the largest exponent of N, as will be defined formally in Eq. (17).

Table 1. Generating matrices for the constituent convolutional codes.

Code description	$G(D)$
Rate 1/2 recursive, parallel	$\left[1, \quad \dfrac{1+D^2}{1+D+D^2}\right]$
Rate 1/2 nonrecursive, outer	$[1+D+D^2, \quad 1+D^2]$
Rate 2/3 recursive, inner	$\begin{bmatrix} 1, & 0, & \dfrac{1+D^2}{1+D+D^2} \\ 0, & 1, & \dfrac{1+D}{1+D+D^2} \end{bmatrix}$

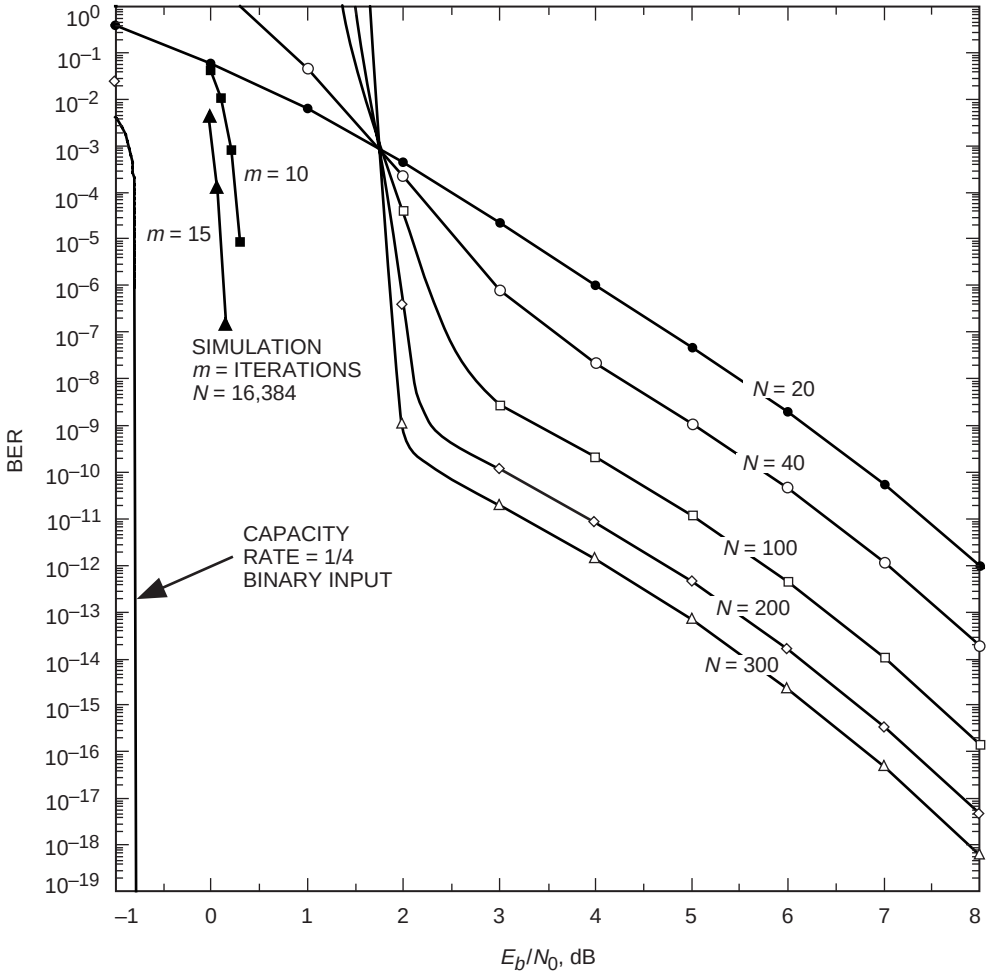

Fig. 2. Analytical bounds for HCCC of Example 1.

performance of the concatenated code improves as the input block size is increased. If the input block size increases, then the sizes of interleavers used in the concatenated code should increase also. When $\alpha_M < 0$, we say that we have "interleaving gain" [6]. The more negative α_M is, the more interleaving gain we can obtain. In order to compute α_M, we proceed as follows: Consider a rate $R = p/n$ convolutional code C with memory ν and its equivalent $(N/R, N - p\nu)$ block code whose codewords are all sequences of length N/R bits of the convolutional code starting from and ending at the zero state. By definition, the codewords of the equivalent block code are concatenations of error events of the convolutional codes. By "error event of a convolutional code" we mean a sequence diverging from the zero state at time $t = 0$ and remerging into the zero state at some discrete time $t > 0$. Let $A^C_{w,h,j}$ be the input–output weight coefficients given that the convolutional code generates j error events with total input weight w and output weight h (see Fig. 3). The $A_{w,h,j}$ actually represents the number of sequences of weight h, input weight w, and the number of concatenated error events j without any gaps between them, starting at the beginning of the block. For N much larger than the memory of the convolutional code, the coefficient $A^C_{w,h}$ of the equivalent block code can be approximated by

$$A^C_{w,h} \sim \sum_{j=1}^{n_M} \binom{N/p}{j} A^C_{w,h,j} \tag{5}$$

where n_M, the largest number of error events concatenated in a codeword of weight h and generated by a weight w input sequence, is a function of h and w that depends on the encoder. The large N assumption permits neglecting the length of error events compared to N, which also implies that the number of ways j input sequences producing j error events can be arranged in a register of length N is $\binom{N/p}{j}$. The ratio N/p derives from the fact that the code has rate p/n and, thus, N bits correspond to N/p input symbols or, equivalently, trellis steps.

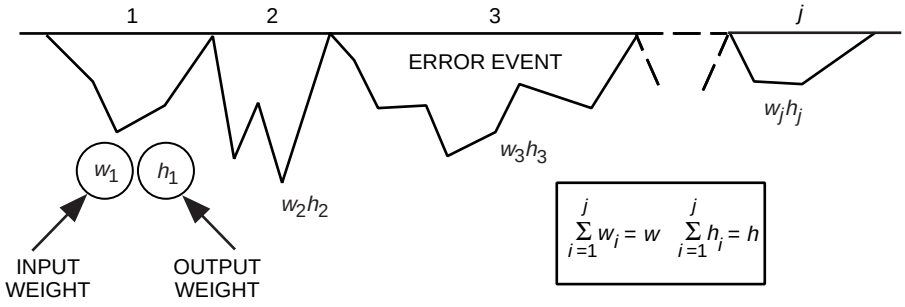

Fig. 3. A code sequence in $A^C_{w,h,j}$.

Let us return now to the block code equivalent to the HCCC. Using Expression (5) with j replaced by n^i for the inner code, j replaced by n^o for the outer code, and, similarly, j replaced by n^p for the parallel code,[3] and noting that $N_2/p = N_1/k_o \triangleq N$, we obtain [9]

$$A^{C_o}_{w,l} \sim \sum_{n^o=1}^{n^o_M} \binom{N}{n^o} A^o_{w,l,n^o} \tag{6}$$

[3] In the following, the superscripts "p","o," and "i" will refer to quantities pertaining to parallel, outer, and inner codes, respectively, and subscripts "m" and "M" will denote "minimum" and "maximum," respectively.

for the outer code and similar expressions for the inner and parallel codes. Then substituting them into Expression (4), we obtain the bit-error probability bound of the hybrid concatenated block code equivalent to the HCCC as

$$P_b(e) \lesssim \sum_{h_1=h_m^p}^{N_1/R_c^p} \sum_{h_2=h_m^i}^{N_2/R_c^i} \sum_{w=w_m}^{N_1} \sum_{l=0}^{N_2} \sum_{n^p=1}^{n_M^p} \sum_{n^o=1}^{n_M^o} \sum_{n^i=1}^{n_M^i} \frac{\binom{N}{n^p}\binom{N}{n^o}\binom{N}{n^i}}{\binom{N_1}{w}\binom{N_2}{l}} A^p_{w,h_1,n^p} A^o_{w,l,n^o} A^i_{l,h_2,n^i}$$

$$\times \frac{w}{N_1} Q\left(\sqrt{2R_c\left(h_1+h_2\right)\frac{E_b}{N_0}}\right) \tag{7}$$

We are interested in large interleaver lengths and, thus, use for the binomial coefficient the asymptotic approximation

$$\binom{N}{n} \sim \frac{N^n}{n!}$$

Substitution of this approximation in Expression (7) gives the bit-error probability bound in the form

$$P_b(e) \lesssim \sum_{h_1=h_m^p}^{N_1/R_c^p} \sum_{h_2=h_m^i}^{N_2/R_c^i} \sum_{w=w_m}^{N_1} \sum_{l=d_f^o}^{N_2} \sum_{n^p=1}^{n_M^p} \sum_{n^o=1}^{n_M^o} \sum_{n^i=1}^{n_M^i} N^{n^p+n^o+n^i-w-l-1} B_{h_1,h_2,w,l,n^p,n^o,n^i}$$

$$\times Q\left(\sqrt{2R_c\left(h_1+h_2\right)\frac{E_b}{N_0}}\right) \tag{8}$$

where

$$B_{h_1,h_2,w,l,n^p,n^o,n^i} = \frac{w!l!}{p^l k_o^w n^p! n^o! n^i!} \frac{w}{k_o} A^o_{w,h_1,n^p} A^o_{w,l,n^o} A^i_{l,h_2,n^i} \tag{9}$$

Using Expression (8), we will obtain some important design rules. The bound, Expression (8), to the bit-error probability can be computed by adding terms of the first two summations with respect to the HCCC weights, $h_1 + h_2$. The IOWC coefficients, which are functions of h_1 and h_2, depend, among other parameters, on N. For large N, and for a given h_1 and h_2, the dominant coefficient corresponding to $h_1 + h_2$ is the one for which the exponent of N is maximum. Define this maximum exponent as

$$\alpha(h_1, h_2) \triangleq \max_{w,l} \{n^p + n^o + n^i - w - l - 1\} \tag{10}$$

where the maximum is over all error events with w, l that produce the output weights h_1 and H_2. Evaluating $\alpha(h_1, h_2)$ is in general not possible without specifying the constituent codes. Thus, we will consider two important cases for which general expressions can be found.

A. The Exponent of N for the Minimum Weight

For large values of E_b/N_0, the performance of the HCCC is dominated by the first terms of the summations in h_1 and h_2, corresponding to the minimum values $h_1 = h_m^p$ and $h_2 = h_m^i$. Noting that n_M^p, n_M^o, and n_M^i are the maximum number of concatenated error events in codewords of the parallel, outer, and inner code of weights h_m^p, l, and h_m^i, respectively, the following holds true:

$$n_M^i = \min\left\{\left\lfloor \frac{h_m^i}{d_f^i}\right\rfloor, \left\lfloor \frac{l}{w_m^i}\right\rfloor\right\} \tag{11}$$

$$n_M^o = \min\left\{\left\lfloor \frac{l}{d_f^o}\right\rfloor, \left\lfloor \frac{w}{w_m^o}\right\rfloor\right\} \tag{12}$$

$$n_M^p = \min\left\{\left\lfloor \frac{h_m^p}{d_f^p}\right\rfloor, \left\lfloor \frac{w}{w_m^p}\right\rfloor\right\} \tag{13}$$

where w_m^i, w_m^o, and w_m^p are minimum weights of input sequences generating codewords with nonpropagating low-output weights for the inner, outer, and parallel encoders, respectively. If the parallel code is nonrecursive, then $n_M^p \le w$, and we obtain $\alpha(h_M^p, h_M^i) \le \max_{w,l}\{n^o + n^i - l - 1\}$. Using the same method for maximization as in [14], we get

$$\alpha\left(h_m^p, h_m^i\right) \le 1 - d_f^o \tag{14}$$

where d_f^o is the minimum Hamming distance of the outer code. This result is similar to one obtained for serial concatenated codes. However, if the parallel code is recursive, then $n_M^p \le \lfloor w/2 \rfloor$ and $\alpha(h_M^p, h_M^i) \le -1 + \max_{w,l}\{n^0 + n^i - l - 1\}$, which, again using the method in [14], can be used to obtain

$$\alpha\left(h_m^p, h_m^i\right) \le -d_f^o \tag{15}$$

The result in Expression (15) shows that the exponent of N corresponding to the minimum weight of HCCC codewords is always negative, thus yielding an interleaver gain at high E_b/N_0. Substitution of the exponent $\alpha(h_m^p, h_m^i)$ into Expression (8) truncated to the first term of the summation in h_1 and h_2 yields

$$\lim_{\frac{E_b}{N_0}\to\infty} P_b(e) \tilde{\le} B_m N^{-d_f^o} Q\left(\sqrt{2R_c\left(h_m^p + h_m^i\right)\frac{E_b}{N_0}}\right) \tag{16}$$

where the constant B_m is independent of N and can be computed from Expression (8) and Eq. (9).

Expression (16) suggests that, for the values of E_b/N_0 and N where the HCCC performance is dominated by its free distance $d_f^{C_H} = h_m^p + h_m^i$, increasing the interleaver length yields a gain in performance. To increase the interleaver gain, one should choose a recursive parallel code and an outer code with large d_f^o. To improve the performance with E_b/N_0, one should choose an inner–parallel code combination such that $h_m^p + h_m^i$ is large. However, as in serial concatenated codes, there are coefficients corresponding to h_1 and h_2 for $h_1 > h_m^p$ and $h_2 > h_m^i$ that may increase with N. Next, we will evaluate the largest exponent of N, defined as

$$\alpha_M \triangleq \max_{h_1,h_2}\{\alpha(h_1,h_2)\} = \max_{w,l,h_1,h_2}\{n^p + n^o + n^i - w - l - 1\} \tag{17}$$

This exponent will allow us to find the dominant contribution to the bit-error probability for $N \to \infty$.

B. The Maximum Exponent of N

We need to treat the cases of nonrecursive and recursive inner encoders separately.

1. The Nonrecursive Inner Encoder. Consider the inner code and its impact on the exponent of N in Eq. (17). For a nonrecursive inner encoder, we have $n_M^i = l$. In fact, every input sequence with weight one generates a nonpropagating low-output-weight error event, so that an input sequence with weight l will generate at most l error events corresponding to the concatenation of l error events of input weight one. Thus, from Eq. (17), we have

$$\alpha_M = \max_{w}\{n^p + n^o - w - 1\}$$

(1) *At least one nonrecursive parallel or outer encoder.* If the parallel encoder is nonrecursive, then $n_M^p = w$, or, if the outer encoder is nonrecursive, then $n_M^o = w$. Therefore, in any case, we have

$$\alpha_M \geq 0$$

and interleaving gain is not allowed.

(2) *Both parallel and outer encoders are recursive.* In [9], it is proved that, for recursive convolutional encoders, the minimum weight of input sequences generating error events is 2. As a consequence, an input sequence of weight w can generate at most $\lfloor w/2 \rfloor$ error events. If both parallel and outer encoders are recursive, then $n_M^p = \lfloor w/2 \rfloor$ and $n_M^o = \lfloor w/2 \rfloor$. In this case, we have

$$\alpha_M = -1$$

and interleaving gain is allowed. This is the same result as for turbo codes [9].

2. The Recursive Inner Encoder. For a recursive inner encoder, we have $n_M^i = \lfloor l/2 \rfloor$. Thus, from Eq. (17), we have

$$\alpha_M = \max_{w,l}\left\{n^p + n^o - w - \left\lfloor \frac{l+1}{2} \right\rfloor - 1\right\}$$

but $l \geq n^o d_f^o$ (note that $d_f^o \geq 2$), so that

$$\alpha_M \leq \max_{w}\left\{n^p - w - \left\lfloor \frac{d_f^o + 1}{2} \right\rfloor\right\}$$

(1) *Nonrecursive parallel encoder.* If the parallel encoder is nonrecursive, we have $n_M^p = w$, thus

$$\alpha_M \leq -\left\lfloor \frac{d_f^o + 1}{2} \right\rfloor$$

and interleaving gain is allowed. This is the same result as for serial concatenated codes [14].

(2) *Recursive parallel encoder.* If the parallel encoder is recursive, we have $n_M^p = \lfloor w/2 \rfloor$. Since $-\lfloor (w+1)/2 \rfloor \leq -1$, we obtain

$$\alpha_M \leq -\left\lfloor \frac{d_f^o + 3}{2} \right\rfloor$$

which shows a higher interleaving gain than for serial concatenated codes. Note that this is a simple upper bound, based only on d_f^o. For a specific structure of the outer code, a tighter bound can be obtained since, if the parallel code is recursive, the input weights $w \geq 2$ should be considered for the outer code. Knowing the structure of the outer code, a tighter bound can be obtained by computing $\alpha_M \leq \max_{w \geq 2} \{ -\lfloor (w+1)/2 \rfloor + \lfloor l_w / d_f^o \rfloor - \lfloor (l_w + 1)/2 \rfloor - 1 \}$, where l_w is the minimum output weight of the outer code for input weight w.

In conclusion, in order to achieve the highest interleaving gain for HCCC, we should select the type of component codes based on the analysis above. Thus, the HCCC should employ

(1) A recursive inner encoder.

(2) A recursive parallel encoder.

(3) An outer encoder that can be either nonrecursive or recursive but that should have large d_f^o.

3. Computation of $h(\alpha_M)$. Next we consider the weight $h(\alpha_M)$ that is the sum of output weights of inner and parallel codes associated with the highest exponent of N:

(1) For even d_f^o, the weight $h(\alpha_M)$ associated with the highest exponent of N is given by

$$h(\alpha_M) = \frac{d_f^o d_{f,eff}^i}{2} + d_{f,eff}^p \tag{18}$$

(2) For odd d_f^o, the value of $h(\alpha_M)$ is given by

$$h(\alpha_M) = \frac{(d_f^o - 3) d_{f,eff}^i}{2} + h_m^{(3)} + d_{f,eff}^p \tag{19}$$

where $h_m^{(3)}$ is the minimum weight of sequences of the inner code generated by weight 3 input sequences. In Eqs. (18) and (19), $d^i_{f,eff}$ and $d^p_{f,eff}$ are the effective free distances of the inner and the parallel codes, respectively. Tables of recursive systematic convolutional codes with maximum effective free distances and maximum $h_m^{(3)}$ are given in [9] and [10].

IV. Iterative Decoding of Hybrid Concatenated Codes

In Sections II and III, we have shown by analytical findings that HCCCs can outperform PCCCs when decoded using an ML algorithm. In practice, however, ML decoding of these codes with large N is an almost impossible task. Thus, to acquire practical significance, the above described codes and analytical bound need to be accompanied by a decoding algorithm of the same order of complexity as that for turbo codes, yet retaining its performance advantages. In this section, we present an iterative algorithm with complexity not significantly higher than that needed to separately decode the three constituent convolutional codes.

As for the iterative decoding of parallel (PCCC) [8,11,17,25] and serial (SCCC) concatenated convolutional codes, the core of the decoding procedure consists of an a posteriori probability (APP) decoding algorithm applied to the constituent convolutional codes. The functionality of the APP decoder to be used for HCCC is slightly different from that needed in the PCCC and even the SCCC decoding algorithm, as we will show shortly. For decoding of the received sequence, we will use the soft-input, soft-output (SISO) APP module described in [12] and [13]. A functional diagram of the iterative decoding algorithm for HCCC is presented in Fig. 4.

We will explain how the algorithm works according to the blocks of Fig. 4. The blocks labeled SISO have two inputs and two outputs. The input labeled $\lambda(c;I)$ represents the reliability of the unconstrained output symbols of the encoder, while that labeled $\lambda(u;I)$ represents the unconstrained input symbols of the encoder. Similarly, the outputs represent the same quantities conditioned to the code constraint as they are evaluated by the APP decoding algorithm in the log domain. As opposed to the iterative decoding algorithm employed for turbo decoding, where the APP algorithm computes only the "extrinsic" information of the input symbols of the encoder conditioned on the code constraint based on the unconstrained reliability of the encoder input symbols, we fully exploit here the potential of the APP algorithm

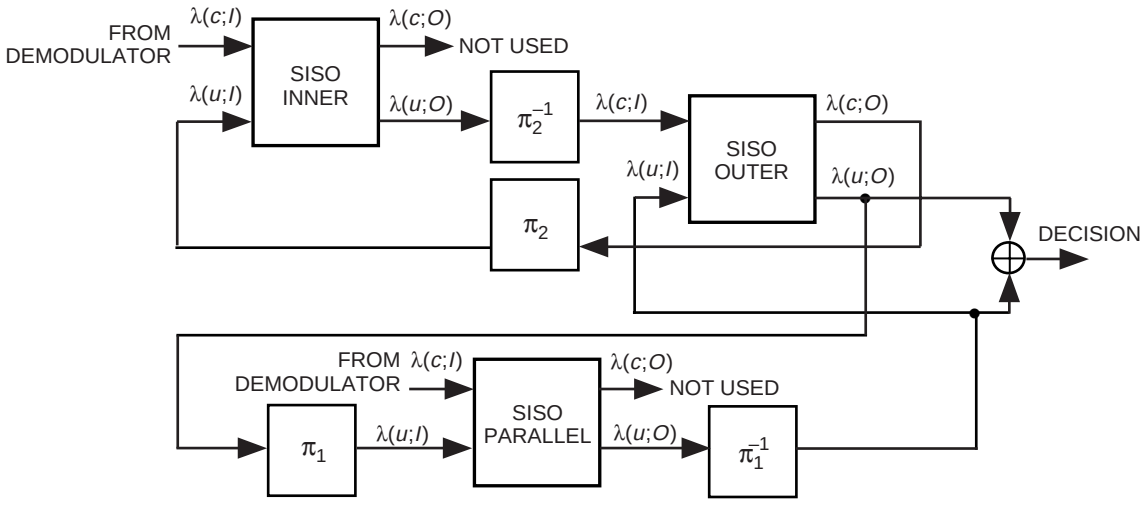

Fig. 4. The iterative decoding algorithm for HCCCs.

at the outer decoder, where all four ports of the SISO are used in the iterative decoder. The SISO APP module updates reliability of both the input and output symbols based on the code constraints. Both outputs of the SISO, i.e., $\lambda(c;O)$ and $\lambda(u;O)$, directly generate the "extrinsic" information required for iterative decoding. So, there is no need to subtract the unconstrained input reliability from the output reliability generated by the APP algorithm.

During the first iteration of the HCCC algorithm,

(1) The block labeled SISO inner is fed with the demodulator soft output, consisting of the reliability of symbols received from the channel, i.e., the received output symbols of the inner encoder. The received reliabilities are processed by the SISO inner module that computes the extrinsic information of the input symbols conditioned on the inner code constraints. This information is passed through the second inverse interleaver (the block labeled π_2^{-1}). As the input symbols of the inner code (after inverse interleaving) correspond to the output symbols of the outer code, they are sent to the SISO module's upper port, which corresponds to the output symbols.

(2) The block labeled SISO parallel is fed with the demodulator soft output, consisting of the reliability of the received symbols from the channel, i.e., the received output symbols of the parallel encoder. The received reliability is processed by the SISO parallel module that computes the extrinsic information of the input symbols conditioned on the parallel code constraints. This information is passed through the first inverse interleaver (the block labeled π_1^{-1}). As the input symbols of the parallel code (after inverse interleaving) correspond to the input symbols of the outer code, they are sent to the SISO module's lower port, which corresponds to the input symbols.

(3) The block labeled SISO outer in turn processes the reliability of the unconstrained output and input symbols received from the SISO inner and parallel modules, respectively, and computes the extrinsic information of both output and input symbols based on the outer code constraints. The extrinsic information of output and input symbols is fed back to the SISO inner and SISO parallel modules in the second iteration as unconstrained input reliabilities.

The reliability of input symbols of the SISO outer module and the reliability of the input symbols of the SISO parallel module will be used in the final iteration to recover the information bits.

A. Bit-by-Bit Iterative Decoding Using the APP SISO Algorithm in Log Domain

For completeness, we briefly describe the SISO algorithm (used for the parallel, inner, and outer convolutional codes) based on the trellis section shown in Fig. 5 for a generic code E with input symbol $\mathbf{u}$ and output symbol $\mathbf{c}$. A detailed description is provided in [13]. Consider an inner code with p_1 input bits and q_1 output bits taking values $\{0,1\}$, a parallel code with p_2 input bits and q_2 output bits taking values $\{0,1\}$, and an outer code with p_3 input bits and q_3 binary outputs $\{0,1\}$. Let the input symbol to the convolutional code $\mathbf{u}_k(e)$ represent the input bits $u_{k,i}(e)$ $i = 1, 2, \cdots, p_m$ on a trellis edge at time k ($m = 1$ for the inner code, $m = 2$ for the parallel code, and $m = 3$ for the outer code), and let the output symbol of the convolutional code $\mathbf{c}_k(e)$ represent the output bits $c_{k,i}(e)$; $i = 1, 2, \cdots, q_m$ ($m = 1$ for the inner code, $m = 2$ for the parallel code, and $m = 3$ for the outer code).

Define the reliability of a bit Z taking values $\{0,1\}$ at time k as

$$\lambda_k[Z;\cdots] \triangleq \log \frac{P_k[Z=1;\cdot]}{P_k[Z=0;\cdot]}$$

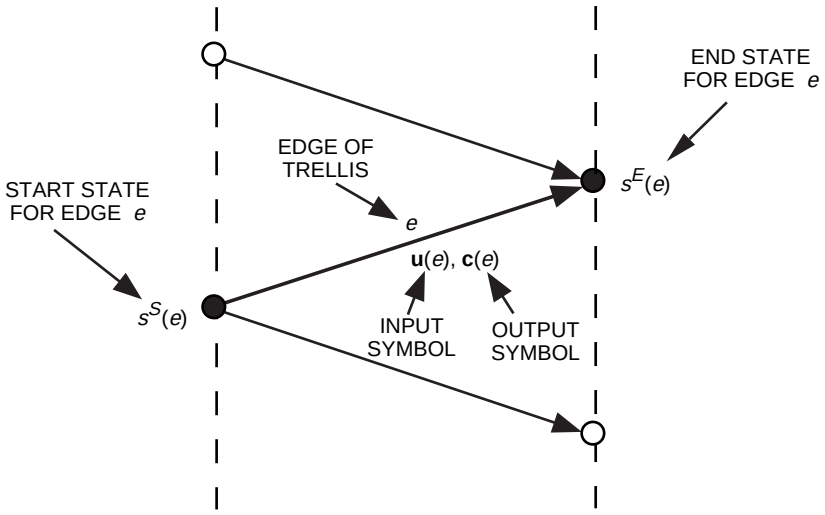

Fig. 5. The trellis section for the code *E*.

The second argument in the brackets, shown as a dot, may represent I, the input, or O, the output, to the SISO. We use the following identity,

$$a = \log\left[\sum_{i=1}^{L} e^{a_i}\right] = \max_i\{a_i\} + \delta(a_1, ..., a_L) \triangleq \max_i{}^*\{a_i\}$$

where $\delta(a_1, \cdots, a_L)$ is a correction term[4] that can be computed using a look-up table. We define the max* operation as a maximization (compare–select) plus a correction term (look-up table). The notation max* was first used in [18] for MAP detection of intersymbol interference channels (see also [8,11,17]). Assume binary PSK communication (QPSK also can be considered, if it uses Gray code mapping, since it is then equivalent to two parallel PSKs). The received samples, $\{y_{k,i}\}$, at the output of the receiver matched filter are normalized such that additive noise samples have unit variance per dimension, i.e., $y_{k,i} = \sqrt{2E_s/N_o}(2c_{k,i} - 1) + n_{k,i}$, which is the assumed channel model. However, the described algorithm works with any other type of normalization if $\lambda_k[C_{k,i}; I]$ in Eq. (20) is redefined accordingly. Without loss of generality, assume all encoders start (at the beginning of the block) at the all-zero state and end at the all-zero state (at the end of the block, when termination is used). For an encoder with memory ν_m, let s represent the state of the encoder, where $s \in \{0, \ldots, 2^{\nu_m}\}$, $m = 1, 2, 3$.

1. The APP SISO Algorithm for the Inner and Parallel Codes. The *forward* and *backward* recursions, respectively, are

$$\alpha_k(s) = \max_{e:s^E(e)=s}{}^* \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{i=1}^{p_m} u_{k,i}(e)\lambda_k\left[U_{k,i}; I\right] + \sum_{i=1}^{q_m} c_{k,i}(e)\lambda_k\left[C_{k,i}; I\right] \right\} + h_{\alpha_k}$$

$$\beta_k(s) = \max_{e:s^S(e)=s}{}^* \left\{ \beta_{k+1}\left[s^E(e)\right] + \sum_{i=1}^{p_m} u_{k+1,i}(e)\lambda_{k+1}\left[U_{k+1,i}; I\right] + \sum_{i=1}^{q_m} c_{k+1,i}(e)\lambda_{k+1}\left[C_{k+1,i}; I\right] \right\} + h_{\beta_k}$$

[4] A. J. Viterbi, "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes," submitted to the *JSAC* issue on *Concatenated Coding Techniques and Iterative Decoding: Sailing Toward Channel Capacity*, January 1998.

with initial values $\alpha_0(s) = 0$, if $s = 0$ (initial zero state), and $\alpha_0(s) = -\infty$, if $s \neq 0$. Similarly, $\beta_n(s) = 0$, if $s = 0$ (final zero state), and $\beta_n(s) = -\infty$, if $s \neq 0$. Recursions are done for $k = 1, \ldots, n-1$, where n represents the total number of trellis steps for an encoder (obviously the value of n can be different for the inner and the parallel codes). The channel reliabilities $\lambda_k[C_{k,i}; I]$ are normalized versions of the observation samples at the output of the matched filter(s). Based on the channel model described above, they can be represented as

$$\lambda_k[C_{k,i}; I] = 2\sqrt{\frac{2E_s}{N_o}} y_{k,i} \tag{20}$$

and h_{α_k} and h_{β_k} are normalization constants. For the hardware implementation of the SISO, these constants are used to prevent buffer overflow. The above operations are similar to the Viterbi algorithm used in the forward and backward directions, except for a correction term that is added when compare–select operations are performed (this is the so-called dual-generalized Viterbi algorithm). If we ignore the correction term, i.e., we replace $\max^*$ with max, we have exactly the Viterbi algorithm in the forward and backward directions (this is also referred to as a bidirectional Viterbi algorithm). There is a small degradation in performance when replacing $\max^*$ with max, which for some cases of turbo codes at low SNRs is roughly 0.5 dB [17].

The *extrinsic bit information* for $U_{k,j}; j = 1, 2 \cdots, p_m;\ m = 1, 2$ can be obtained from

$$\lambda_k(U_{k,j}; O) = \max^*_{e:u_{k,j}(e)=1} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{\substack{i=1 \\ i\neq j}}^{p_m} u_{k,i}(e)\lambda_k[U_{k,i}; I] + \sum_{i=1}^{q_m} c_{k,i}(e)\lambda_k\left[C_{k,i}(e); I\right] + \beta_k\left[s^E(e)\right] \right\}$$

$$- \max^*_{e:u_{k,j}(e)=0} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{\substack{i=1 \\ i\neq j}}^{p_m} u_{k,i}(e)\lambda_k\left[U_{k,i}; I\right] + \sum_{i=1}^{q_m} c_{k,i}(e)\lambda_k\left[C_{k,i}(e); I\right] + \beta_k\left[s^E(e)\right] \right\}$$

At the first iteration, all input reliabilities $\lambda_k[U_{k,i}; I]$ are set to zero. The inner and parallel SISOs compute the extrinsic information $\lambda_k(U_{k,j}; O)$ from the above equations and provide it to the outer SISO. After the first iteration, the inner and parallel SISOs accept the extrinsics from the outer SISO (to be described next) as the reliabilities of input bits of the inner and parallel encoders, together with the external observations from the channel for the computation of the new extrinsic information $\lambda_k(U_{k,j}; O)$ for the input bits. These are then provided to the outer SISO module.

2. The SISO Algorithm for the Outer Code. The *forward* and *backward* recursions, respectively, are

$$\alpha_k(s) = \max^*_{e:s^E(e)=s} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{i=1}^{p_3} u_{k,i}(e)\lambda_k\left[U_{k,i}; I\right] + \sum_{i=1}^{q_3} c_{k,i}(e)\lambda_k\left[C_{k,i}; I\right] \right\} + h_{\alpha_k}$$

$$\beta_k(s) = \max^*_{e:s^S(e)=s} \left\{ \beta_{k+1}\left[s^E(e)\right] + \sum_{i=1}^{p_3} u_{k+1,i}(e)\lambda_{k+1}\left[U_{k+1,i}; I\right] + \sum_{i=1}^{q_3} c_{k+1,i}(e)\lambda_{k+1}\left[C_{k+1,i}; I\right] \right\} + h_{\beta_k}$$

The *extrinsic information* for $C_{k,j}; j = 1, 2, \cdots, q_3$, can be obtained from

$$\lambda_k(C_{k,j};O) = \max_{e:c_{k,j}(e)=1}{}^{*} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{i=1}^{p_3} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{\substack{i=1\\ i\neq j}}^{q_3} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] + \beta_k\left[s^E(e)\right] \right\}$$

$$- \max_{e:c_{k,j}(e)=0}{}^{*} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{i=1}^{p_3} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{\substack{i=1\\ i\neq j}}^{q_3} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] + \beta_k\left[s^E(e)\right] \right\}$$

with initial values $\alpha_0(s) = 0$ if $s = 0$, and $\alpha_0(s) = -\infty$ if $s \neq 0$. Similarly, $\beta_n(s) = 0$ if $s = 0$, and $\beta_n(s) = -\infty$ if $s \neq 0$, where h_{α_k} and h_{β_k} are again normalization constants that for a hardware implementation of the SISO are used to prevent the buffer overflow.

The *extrinsic information* for $U_{k,j}; j = 1, 2, \cdots, p_3$, can be obtained from

$$\lambda_k(U_{k,j};O) = \max_{e:u_{k,j}(e)=1}{}^{*} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{\substack{i=1\\ i\neq j}}^{p_3} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{i=1}^{q_3} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] + \beta_k\left[s^E(e)\right] \right\}$$

$$- \max_{e:u_{k,j}(e)=0}{}^{*} \left\{ \alpha_{k-1}\left[s^S(e)\right] + \sum_{\substack{i=1\\ i\neq j}}^{p_3} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{i=1}^{q_3} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] + \beta_k\left[s^E(e)\right] \right\}$$

The final decision can be obtained from the bit-reliability computation of $U_{k,j}; j = 1, 2, \cdots, p_3$, as

$$L_{k,j} = \lambda_k\left(U_{k,j};O\right) + \lambda_k\left[U_{k,j};I\right]$$

and then passing it through a hard limiter.

In the iterative decoding, the outer SISO accepts the extrinsics from the inner SISO and parallel SISO as input reliabilities of coded bits and input bits, respectively, of the outer encoder. For the outer SISO, there is no external observation from the channel. The outer SISO uses the input reliabilities for calculation of new extrinsics $\lambda_k(C_{k,j};O)$ for coded bits and $\lambda_k(U_{k,j};O)$ for the input information bits. These are then provided to the inner SISO and parallel SISO modules.

3. The SISO Algorithm for a Very Short Block Code as an Outer Code. When the outer code is a block code, one can obtain trellis representation of the block code and use the SISO algorithm described in Section IV.A.2. However, if the block code (e.g., parity check code, Hamming code, or BCH code) is very short, one can use the simplified version of the SISO algorithm. In this case, there is no need for the forward and backward recursions. Actually, we can have a one-state trellis section, where the number of edges is equal to the number of codewords of the block code under consideration. Then we have the following brute-force-type algorithm: Assume a (q,p) block code with input $\mathbf{u}$ and output codewords $\mathbf{c}$. Note the input block size of the HCCC (or SCCC) in this case can be an integer multiple of p, where p is small, say $p \leq 8$.

The *extrinsic information* for $C_{k,j}; j = 1, 2, \cdots, q$, can be obtained from

$$\lambda_k(C_{k,j};O) = \max_{e:c_{k,j}(e)=1}{}^{*} \left\{ \sum_{i=1}^{p} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{\substack{i=1 \\ i\neq j}}^{q} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] \right\}$$

$$- \max_{e:c_{k,j}(e)=0}{}^{*} \left\{ \sum_{i=1}^{p} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{\substack{i=1 \\ i\neq j}}^{q} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] \right\}$$

The *extrinsic information* for $U_{k,j}; j = 1, 2, \cdots, p$, can be obtained from

$$\lambda_k(U_{k,j};O) = \max_{e:u_{k,j}(e)=1}{}^{*} \left\{ \sum_{\substack{i=1 \\ i\neq j}}^{p} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{i=1}^{q} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] \right\}$$

$$- \max_{e:u_{k,j}(e)=0}{}^{*} \left\{ \sum_{\substack{i=1 \\ i\neq j}}^{p} u_{k,i}(e)\lambda_k\left[U_{k,i};I\right] + \sum_{i=1}^{q} c_{k,i}(e)\lambda_k\left[C_{k,i};I\right] \right\}$$

V. Special Case of a Hybrid Concatenated Code When the Outer Code is Repetition Code

When the outer code is a repetition code, the hybrid concatenated code reduces to a so-called "repetitive turbo code," as suggested by M. Bingeman and A. K. Khandani in [20], where they compared by simulation the suggested scheme with parallel concatenation of three convolutional codes.

Here we can apply the analytical results and the decoder described in Section IV for hybrid concatenated codes to this special case.[5] However, for the repetition code, the SISO algorithm reduces to a simpler operation, as will be explained in Section V.A.

For a rate $1/q$ repetition code concatenated N times, we have $A_{w,l}^{C_o} = \binom{N}{w}; l = wq$, and zero otherwise, where N is the input block size in bits. Using Expression (5), we obtain

$$P_b(e) \leq \sum_{h_1=h_m^p}^{N/R_c^p} \sum_{h_2=h_m^i}^{Nq/R_c^i} \sum_{w=w_m}^{N} \frac{A_{w,h_1}^{C_p} \times A_{wq,h_2}^{C_i}}{\binom{Nq}{wq}} \frac{w}{N} Q\left(\sqrt{2R_c\left(h_1+h_2\right)\frac{E_b}{N_0}}\right) \tag{21}$$

In this special case, there is no need to use the interleaver π_1. Applying the results in Section II to this special case, we obtain α_M and $h(\alpha_M)$ for the preferred case where both convolutional codes are recursive. Using the tighter upper bound on α_M described in Section III.B.2, or just Eq. (17), and noting that $n^o = w$, $n_M^p = \lfloor w/2 \rfloor$, $n_M^i = \lfloor l/2 \rfloor$, and $l = wq$, and after maximization, we obtain

$$\alpha_M = -q$$

[5] A different iterative decoder that uses repetition and a "recombination algorithm" that requires threshold comparisons was suggested by Bingeman and Khandani.

Next we obtain the weight $h(\alpha_M)$, which is the sum of the minimum output weights of the inner and parallel codes associated with the highest exponent of N as

$$h(\alpha_M) = q\ d^i_{f,eff} + d^p_{f,eff} \tag{22}$$

where, in Eq. (22), $d^i_{f,eff}$ and $d^p_{f,eff}$ are the effective free distances of the inner and parallel codes, respectively. This is due to a single error event with input weight 2 for the parallel code and q error events, each of input weight 2, for the inner code. For this scheme, as for turbo codes, input data are sent once. This can be done by sending the systematic bits of the parallel encoder and not sending the systematic bits of the inner encoder. It is interesting to note that exactly the same results for α_M and $h(\alpha_M)$ can be obtained when we consider parallel concatenation of $q+1$ convolutional codes (multiple turbo codes) when q of them are identical.

A. The SISO Algorithm for Rate 1/q Repetition Outer Code

Rate $1/q$ repetition codes can be viewed as a one-state encoder with a trellis section, as shown in Fig. 6. The SISO algorithm described for the outer code can now be simplified by setting $\alpha_k(s) = 0$ and $\beta_k(s) = 0$. Since there are only one state and two edges corresponding to input bits 0 and 1, there is no need for the forward recursion, the backward recursion, or the max* operation. If we denote the input bit by u_k and the output bits by $c_k, j;\ j = 1, \ldots, q$, then we obtain the following results. This is a simplified version of the SISO algorithm for a very short block code, described in Section IV.A.3.

The *extrinsic information* for $C_{k,j}; j = 1, 2, \cdots, q$, can be obtained from

$$\lambda_k(C_{k,j}; O) = \lambda_k\left[U_k; I\right] + \sum_{\substack{i=1 \\ i \neq j}}^{q} \lambda_k\left[C_{k,i}; I\right]$$

The *extrinsic information* for U_k can be obtained from

$$\lambda_k(U_{k,j}; O) = \sum_{i=1}^{q} \lambda_k[C_k; I]$$

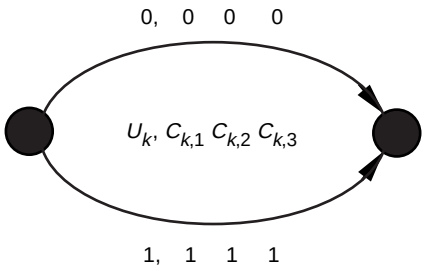

Fig. 6. The trellis section of repetition code, $q = 3$.

VI. Self-Concatenated Code

Consider a self-concatenated code as shown in Fig. 7. This code can be considered as a special case of a hybrid concatenated code where the outer code C_o is a rate $1/q$ repetition code and the parallel code is a rate 1, one-state code (no code). Since only one nontrivial code is used, we call this structure "self-concatenated." If C_i is systematic recursive convolutional code, the systematic bits of the code are not transmitted. Various overall code rates can be obtained by puncturing the parity bits of C_i.

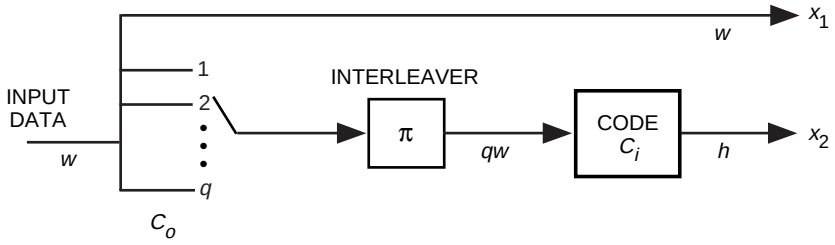

Fig. 7. A self-concatenated code.

For the rate $1/q$ repetition code and its N concatenations, we have $A_{w,l}^{C_o} = \binom{N}{w}; l = wq$, and zero otherwise. For the rate 1, one-state parallel code (no code), we have $A_{w,h_1}^{C_p} = \binom{N}{w}; h_1 = w$, and zero otherwise. Using Expression (4), we obtain

$$P_b \leq \sum_{h=h_m}^{Nq/R_c^i} \sum_{w=1}^{N} \frac{w}{N} \frac{\binom{N}{w} A_{qw,h}^{C_i}}{\binom{qN}{qw}} Q\left(\sqrt{\frac{2R_c E_b}{N_0}(h+w)}\right) \tag{23}$$

Let us consider a special case of a self-concatenated code. If the interleaver is split into q equal sections corresponding to the original data and its $q-1$ duplicates, and we send the data and its $q-1$ permuted versions sequentially through a single convolutional code, then the overall code will be equivalent to a multiple turbo code (parallel concatenation of q codes). The special case of $q = 2$ with a structured interleaver, which does not require trellis termination, was proposed by Berrou [16]. This is just another way to implement turbo codes with two identical constituent convolutional codes using a single convolutional encoder. Now let us go back to our proposed self-concatenated code where we do not partition the interleaver into q interleavers. This allows us to obtain a better fixed interleaver of size qN, using a spread interleaver [5]. However, for the analysis in the next subsection, we use a uniform interleaver [6], which shows the same interleaving gain as for turbo codes.

A. The Maximum Exponent of N for the Self-Concatenated Code

Using Eq. (17), we obtain $\alpha_M = \max_{w,h}\{w + n^i - qw - 1\}$, where n^i is the number of concatenated error events in code C_i. If we use a nonrecursive convolutional encoder C_i, as in Fig. 7, we have $n^i \leq qw$. In this case, $\alpha_M \geq 0$. Thus, we have *no interleaving gain*. However, for the recursive convolutional encoder C_i, the minimum weight of input sequences generating error events is 2. As a consequence, an input sequence of weight qw can generate at most $n^i = \lfloor qw/2 \rfloor$ error events. In this case, the exponent of N is negative, and we have an *interleaving gain*. For $q = 2$, the maximum exponent of N is equal to -1, and the minimum output weight is $h + w = d_{f,eff} + 1$. For $q = 3$, the maximum exponent of N is equal to -2, and the minimum output weight is $h + w = h_m^{(3)} + 1$. However, if $h = h_m^{(3)} = \infty$, then the minimum output weight is $h + w = 3d_{f,eff} + 2$. We also could use the upper bound on α_M of Section III.B.2 for nonrecursive parallel code to obtain the previous results by noting that $d_f^o = q$.

An iterative decoder for the self-concatenated code, which we call a "self-iterative decoder" (since SISO for repetition code requires only q adders), is shown in Fig. 8 for $q = 3$.

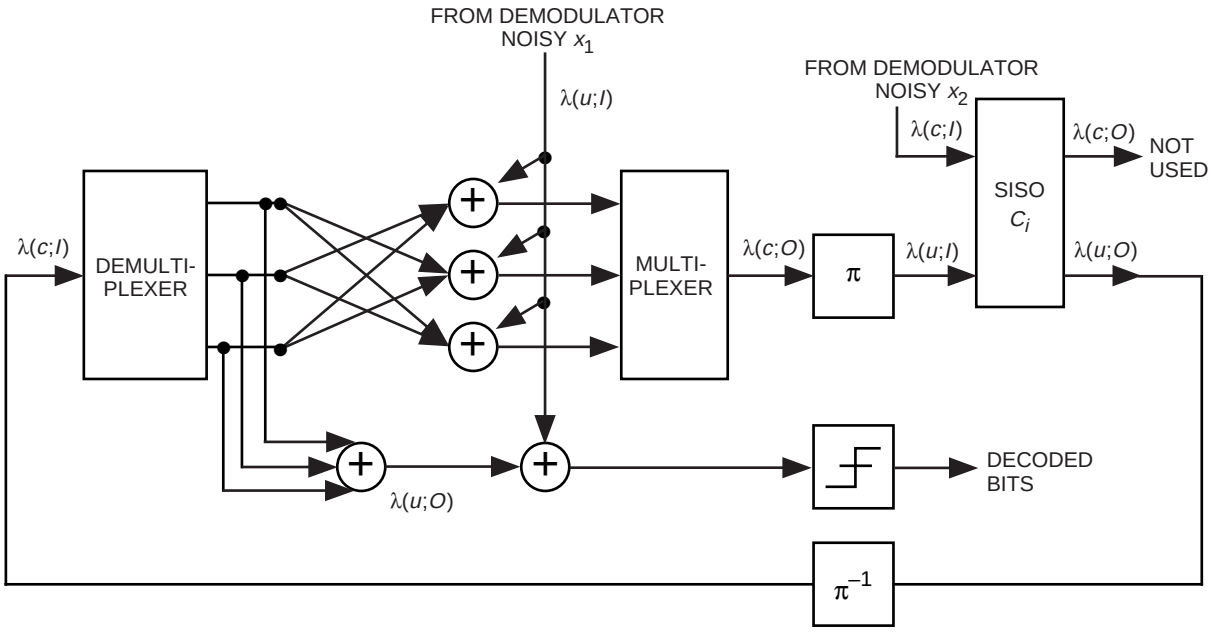

Fig. 8. A self-iterative decoder.

VII. Simulation Results

The HCCC code proposed in Example 1, namely a rate 1/4 HCCC with three four-state constituent convolutional codes, was simulated over an AWGN channel for an input block of N=16,384 bits. The iterative decoding scheme described in Section IV was used for simulations. Simulation results are compared to analytic bounds (where the bounds are computed for much smaller block sizes) in Fig. 2. The bit-error probability versus the number of iterations, for a given E_b/N_0 in dB as a parameter, is shown in Fig. 9(a). The same results are also plotted as bit-error probability versus E_b/N_0 in dB, for a given number of iterations as a parameter, in Fig. 9(b). As is seen from Fig. 9(a) at E_b/N_0=0.1 dB, a bit-error probability of 10^{-5} can be achieved with 19 iterations. Thus, at this bit-error probability, the performance is within 0.9 dB from the Shannon limit for rate 1/4 codes, over binary input AWGN channels. An optimum rate 1/4 PCCC (turbo code) with two four-state codes at 10^{-5} requires E_b/N_0=0.4 dB with 15 iterations (there is a very small improvement if we use 18 iterations instead of 15 for PCCC) and an input block of N=16,384 bits. The main advantage of HCCC over PCCC, as can be seen from Fig. 9(b), occurs at low bit-error probabilities (less than 10^{-6}). Even if we increase the number of states of the PCCC constituent codes to 8 states, the HCCC with three four-state codes outperforms the PCCC at low bit-error rates. This also was predicted by our analysis. At high bit-error probabilities, HCCC and SCCC have similar performances (for the simulation performance of rate 1/4 SCCC with two four-state codes and an input block of N=16,384 bits, see [14]). However, at very low bit-error probabilities, HCCC outperforms SCCC since, based on our analysis, the interleaving gain is N^{-4} for HCCC and N^{-3} for SCCC (a factor of 16,384), while $h(\alpha_M) = 11$ in both cases. To obtain the results in Figs. 9(a) and 9(b), 5×10^8 random bits were simulated for each point.

A. Simulation Results for a Self-Concatenated Code

Consider a rate 1/4 self-concatenated code with a four-state recursive convolutional code $G(D) = [1, (1 + D^2)/(1 + D + D^2)]$, $q = 3$, and an input block of $N = 256$ and 1024 bits, as shown in Fig. 10.

The simulation result for this code, using the iterative decoder in Fig. 8, is shown in Fig. 11. One use of SISO per input block was considered as one iteration. Interleavers with spread 11 and 23 were used for $N = 256$ and $N = 1024$, respectively, in the simulations. For $N = 256$ and an interleaver with spread 11, the minimum output weight of 26 was obtained for all possible input sequences ($N = 256$) of weight 1, 2, 3. This minimum output weight occurred for input weight 2 only once.

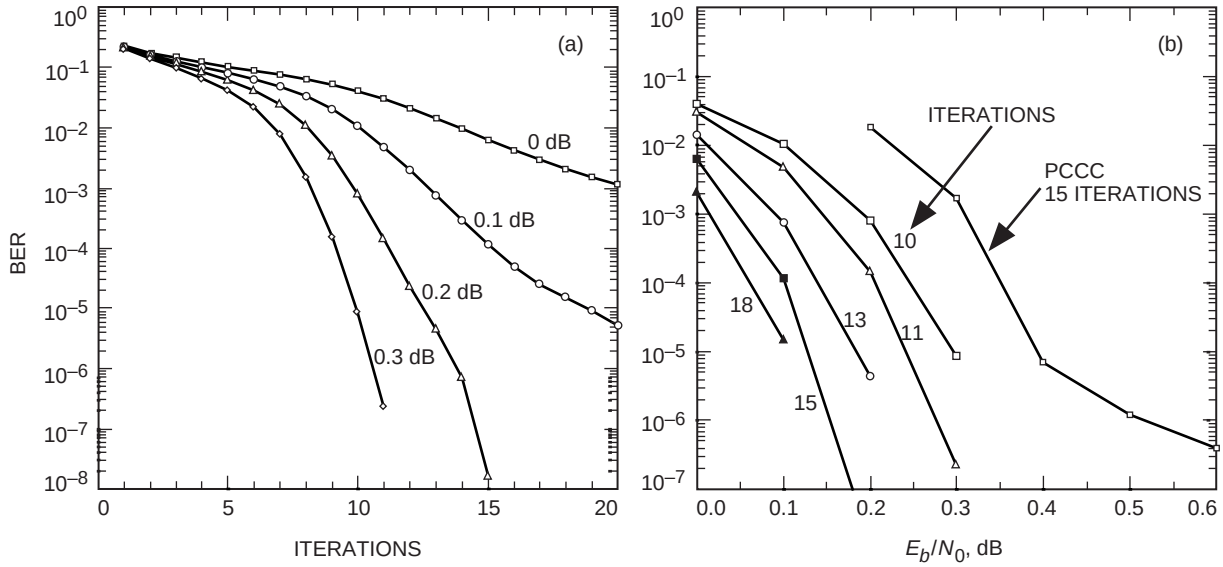

Fig. 9. Simulation results for the HCCC with $N = 16{,}384$ bits: (a) BER versus number of iterations at different bit SNRs and (b) BER versus E_b/N_0 at different numbers of iterations.

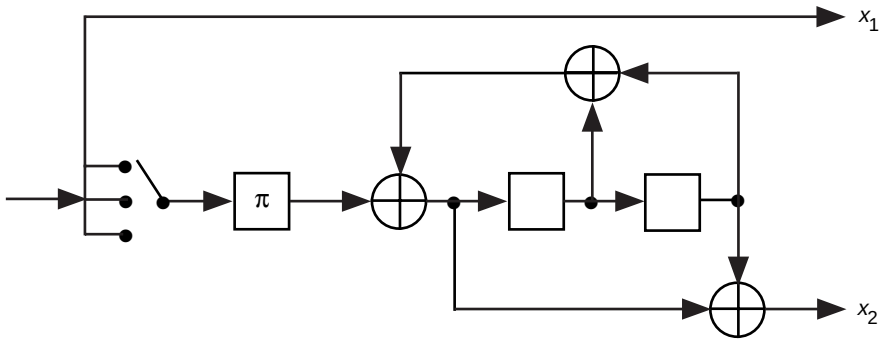

Fig. 10. Simulation results for rate 1/4 self-concatenated code with $q = 3$ and $N = 256$ and 1024 bits.

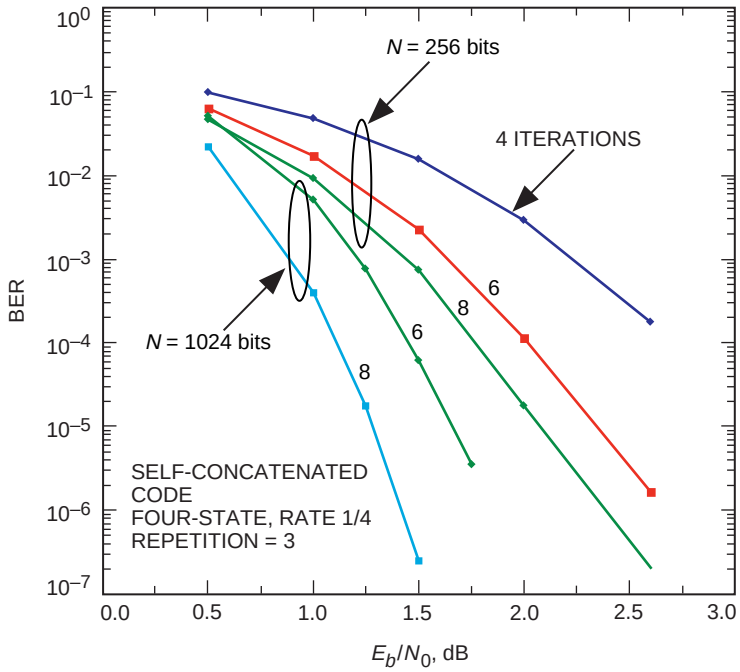

Fig. 11. Simulation results for rate 1/4 self-concatenated code with $q = 3$ and $N = 256$ and 1024 bits.

Acknowledgments

The authors would like to thank Sergio Benedetto and Guido Montorsi of the Politecnico di Torino, Italy, for their helpful comments and suggestions.

References

[1] G. D. Forney, Jr., *Concatenated Codes*, Cambridge, Massachusetts: M.I.T., 1966.

[2] R. H. Deng and D. J. Costello, "High Rate Concatenated Coding Systems Using Bandwidth Efficient Trellis Inner Codes," *IEEE Transactions on Communications*, vol. COM-37, no. 5, pp. 420–427, May 1989.

[3] J. Hagenauer and P. Hoeher, "Concatenated Viterbi Decoding," *Proceedings of Fourth Joint Swedish–Soviet Int. Workshop on Information Theory*, Gotland, Sweden, Studenlitteratur, Lund, pp. 29–33, August 1989.

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

[5] D. Divsalar and F. Pollara, "Turbo Codes for PCS Applications," *Proceedings of IEEE ICC'95*, Seattle, Washington, June 1995.

[6] S. Benedetto and G. Montorsi, "Unveiling Turbo-Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Transactions on Information Theory*, vol. 43, no. 2, March 1996.

[7] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, pp. 284–287, March 1974.

[8] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," *Proceedings of ICC'95*, Seattle, Washington, pp. 1009–1013, June 1995.

[9] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," *IEEE Transactions on Communications*, vol. 44, no. 5, pp. 591–600, May 1996.

[10] D. Divsalar and R. J. McEliece, "The Effective Free Distance of Turbo Codes," *IEE Electronic Letters*, vol. 32, no. 5, pp. 445–446, February 29, 1996.

[11] S. S. Pietrobon, "Implementation and Performance of a Serial MAP Decoder for Use in an Iterative Turbo Decoder," *Proceedings of 1995 IEEE International Symposium on Information Theory*, Whistler, British Columbia, Canada, p. 471, September 1995.

[12] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-Output Decoding Algorithms for Continuous Decoding of Parallel Concatenated Convolutional Codes," *Proceedings of ICC'96*, Dallas, Texas, June 1996.

[13] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "A Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes," *The Telecommunications and Data Acquisition Progress Report 42-127, July–September 1996*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–20, November 15, 1996.
http://tda.jpl.nasa.gov/tda/progress_report/42-127/127H.pdf

[14] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *The Telecommunications and Data Acquisition Progress Report 42-126, April–June 1996*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–26, August 15, 1996.
http://tda.jpl.nasa.gov/tda/progress_report/42-126/126D.pdf

[15] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-Input Soft-Output Building Blocks for the Construction and Distributed Iterative Decoding of Code Networks," *European Transactions on Telecommunications*, invited paper to special issue, to appear in January–February 1998.

[16] C. Berrou and M. Jezequel, "Frame-Oriented Convolutional Turbo Codes," *Electronics Letters*, vol. 32, no. 15, July 18, 1996.

[17] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes," *The Telecommunications and Data Acquisition Progress Report 42-124, October–December 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 63–87, February 15, 1996.
http://tda.jpl.nasa.gov/tda/progress_report/42-124/124G.pdf

[18] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced Complexity Symbol Detectors With Parallel Structures for ISI Channels," *IEEE Transactions on Communications*, vol. 42, no. 2-4, pt. 3, pp. 1661–1671, February–April 1994.

[19] R. M. Tanner, *Error-Correcting Coding System*, U.S. Patent 4,295,218, October 13, 1981.

[20] M. Bingeman and A. K. Khandani, "Repetitive Turbo-Codes for High Redundancy Coding," *Proceedings of the 1997 Conference on Information Sciences and Systems*, Baltimore, Maryland, March 1997.

[21] J. H. Lodge, R. J. Young, P. Hoeher, and J. Hagenauer, "Separable MAP 'Filters' for the Decoding of Product Codes and Concatenated Codes," *Proceedings of ICC'93*, Geneva, Switzerland, pp. 1740–1745, May 1993.

[22] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara, "Transfer Function Bounds on the Performance of Turbo Codes," *The Telecommunications and Data Acquisition Progress Report, April–June 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 44–55, August 15, 1995.
http://tda.jpl.nasa.gov/tda/progress_report/42-122/122A.pdf

[23] S. Dolinar and D. Divsalar, Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations," *The Telecommunications and Data Acquisition Progress Report, April–June 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 56–65, August 15, 1995.
http://tda.jpl.nasa.gov/tda/progress_report/42-122/122B.pdf

[24] L. C. Perez, J. Seghers, and D. J. Costello, Jr., "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Transactions on Information Theory*, vol. IT-42, pp. 1698–1709, November 1996.

[25] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.