Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding

Sergio Benedetto, *Fellow, IEEE*, Dariush Divsalar, *Fellow, IEEE*,
Guido Montorsi, *Member, IEEE*, and Fabrizio Pollara, *Member, IEEE*

***Abstract*—A serially concatenated code with interleaver consists of the cascade of an *outer* encoder, an interleaver permuting the outer codewords bits, and an *inner* encoder whose input words are the permuted outer codewords. The construction can be generalized to h cascaded encoders separated by $h-1$ interleavers. We obtain upper bounds to the *average* maximum-likelihood bit error probability of serially concatenated block and convolutional coding schemes. Then, we derive design guidelines for the outer and inner encoders that maximize the *interleaver* gain and the asymptotic slope of the error probability curves. Finally, we propose a new, low-complexity iterative decoding algorithm. Throughout the paper, extensive comparisons with parallel concatenated convolutional codes known as "turbo codes" are performed, showing that the new scheme can offer superior performance.**

***Index Terms*— Concatenated codes, iterative decoding, serial concatenation, turbo codes.**

NOMENCLATURE

CC	Constituent Code.
PCCC	Parallel Concatenated Convolutional Code.
PCBC	Parallel Concatenated Block Code.
SCC	Serially Concatenated Code.
SCBC	Serially Concatenated Block Code.
SCCC	Serially Concatenated Convolutional Code.
ML	Maximum Likelihood.
IOWEF	Input–Output Weight-Enumerating Function.
CWEF	Conditional Weight-Enumerating Function.
SISO	Soft Input Soft Output module.
SW-SISO	Sliding Window—Soft Input Soft Output module.
LLR	Log-Likelihood Ratio.
MAP	Maximum *a posteriori*.

Manuscript received July 26, 1996; revised November 1, 1997. This work was supported in part by NATO under Research Grant CRG 951208. The work of S. Benedetto and G. Montorsi was supported by Italian National Research Council (CNR) under "Progetto Finalizzato Trasporti (Prometheus)," by MURST (Progetto 40% "Comunicazioni con Mezzi Mobili"), and by Qualcomm. The research described in this paper was performed in part at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, under Contract with the National Aeronautics and Space Administration (NASA). The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Ulm, Germany, June 29–July 4, 1997.

S. Benedetto and G. Montorsi are with Dipartimento di Elettronica, Politecnico di Torino, Italy.

D. Divsalar and F. Pollara are with Jet Propulsion Laboratory, Pasadena, CA 91109 USA.

Publisher Item Identifier S 0018-9448(98)02352-9.

I. INTRODUCTION

IN his goal to find a class of codes whose probability of error decreased exponentially at rates less than capacity, while decoding complexity increased only algebraically, Forney [1] arrived at a solution consisting of the multilevel coding structure known as *concatenated code*. It consists of the cascade of an *inner* code and an *outer* code, which, in Forney's approach, would be a relatively short inner code (typically, a convolutional code) admitting simple maximum-likelihood (ML) decoding, and a long high-rate algebraic nonbinary Reed–Solomon outer code equipped with a powerful algebraic error-correction algorithm, possibly using reliability information from the inner decoder.

Initially motivated only by theoretical research interests, concatenated codes have since then evolved as a standard for those applications where very high coding gains are needed, such as (deep-) space applications and many others. Alternative solutions for concatenation have also been studied, such as using a trellis-coded modulation scheme as inner code [2], or concatenating two convolutional codes [3]. In the latter case, the inner Viterbi decoder employs a soft-output decoding algorithm to provide soft-input decisions to the outer Viterbi decoder. An interleaver was also proposed between the two encoders to separate bursts of errors produced by the inner decoder.

We find then, in a "classical" concatenated coding scheme, the main ingredients that formed the basis for the invention of "turbo codes" [4], namely two, or more, *constituent* codes (CC's) and an *interleaver*. The novelty of turbo codes, however, consists of the way they use the interleaver, which is embedded into the code structure to form an overall concatenated code with very large block length, and in the proposal of a parallel concatenation to achieve a higher rate for given rates of CC's. The latter advantage is obtained using systematic CC's and not transmitting the information bits entering the second encoder. In the following, we will refer to turbo codes as *parallel concatenated convolutional codes* (PCCC's). The so-obtained codes have been shown to yield very high coding gains at bit error probabilities in the range $10^{-5} - 10^{-7}$; in particular, low bit error probabilities can be obtained at rates well beyond the channel cutoff rate, which had been regarded for long time as the "practical" capacity. Quite remarkably, this performance can be achieved by a relatively simple iterative decoding technique whose computational complexity is comparable to that needed to decode the two CC's.

In this paper, we consider the serial concatenation of interleaved codes or *serially concatenated codes* (SCC's), called SCBC or SCCC according to the nature of CC's, that can be block (SCBC) or convolutional (SCCC) codes. For this class of codes, we obtain analytical upper bounds to the performance of an ML decoder, propose design guidelines leading to the optimal choice of CC's that maximize the *interleaver gain* and the asymptotic code performance, and present a new iterative decoding algorithm yielding results close to capacity limits with limited decoding complexity. Preliminary results have appeared in [5] and [6]. Extensive comparisons with turbo codes of the same complexity and decoding delay are performed.

With this embodiment of results, we believe that SCCC can be considered as a valid, in some cases superior, alternative to turbo codes.

In Section II, we derive analytical upper bounds to the bit error probability of both SCBC's and SCCC's, using the concept of "uniform interleaver" that decouples the output of the outer encoder from the input of the inner encoder. In Section III, we propose design rules for SCCC's through an asymptotic approximation of the bit error probability bound assuming long interleavers or large signal-to-noise ratios. In Section III, we compare serial and parallel concatenations of block and convolutional codes in terms of maximum-likelihood analytical upper bounds. Section V is devoted to the presentation of a new iterative decoding algorithm and to its application to some significant codes. Performance comparison between SCCC's and PCCC's under suboptimum iterative decoding algorithms are presented in Section IV.

II. Analytical Bounds to the Performance of Serially Concatenated Codes

A. A Union Bound to the Bit Error Probability and Some General Warnings

Consider an (n, k) linear block code and its *Input–Output Weight Enumerating Function* (IOWEF), defined as

$$A(W, H) \triangleq \sum_{w=0}^{k}\sum_{h=0}^{n} A_{w,h}W^wH^h = \sum_{w=0}^{k} W^wA(w, H) \tag{1}$$

where $A_{w,h}$ represents the number of codewords with weight h generated by information words of weight w. In (1), we have also implicitly defined the *conditional* weight enumerating function (CWEF)

$$A(w, H) \triangleq \sum_{h=0}^{n} A_{w,h}H^h \tag{2}$$

as the function that enumerates the weight distribution of codewords generated by information words of a given weight w.

A linear block (or convolutional) code possesses the *uniform error property* [7], stating that its word and bit error probability performance can be evaluated under the assumption that the all-zero codeword has been transmitted. Assume then that the all-zero codeword $\boldsymbol{x}_0$ has been transmitted, and define the *pairwise* error event $e_{0h}(w)$ as the event in which the likelihood of a codeword with weight h and generated by an information word of weight w is higher than that of the all-zero codeword $\boldsymbol{x}_0$.

Using the union bound, the bit error probability under ML soft decoding for binary phase-shift keying (PSK) (or binary pulse amplitude modulation (PAM)) transmission over an additive white Gaussian noise channel with two-sided noise power spectral density $N_0/2$ can be upper-bounded as

$$\begin{aligned} P_b(e) &\le \sum_{h=1}^{n}\sum_{w=1}^{k} P_b[e|\boldsymbol{x}_0,\, e_{0h}(w)]P[e_{0h}(w)] \\ &= \frac{1}{2}\sum_{h=1}^{n}\sum_{w=1}^{k} \frac{w}{k}A_{w,h}\text{erfc}\left(\sqrt{\frac{hR_cE_b}{N_0}}\right) \\ &= \frac{1}{2}\sum_{h=1}^{n} B_h^{(b)}\text{erfc}\left(\sqrt{\frac{hR_cE_b}{N_0}}\right) \end{aligned} \tag{3}$$

where R_c is the code rate, E_b is the energy per *information* bit,[1] and where we have defined the *bit error multiplicity*

$$B_h^{(b)} \triangleq \sum_{w=1}^{k} \frac{w}{k}A_{w,h}. \tag{4}$$

Expressions (3) and (4) suggest that two ways can be followed to improve the bit error probability performance: the first, leading to the more traditional concept of good (and asymptotically good) codes, tries to increase the first, more significant weights h in (3); the second, forming the basis of turbo codes and also of serially concatenated codes, aims at reducing the bit error multiplicities (4). To quote Forney's 1995 Shannon lecture:

> *Rather than attacking error exponents, turbo codes attack multiplicities, turning conventional wisdom on its head.*

A more compact, but looser, upper bound, can be obtained from (3) using the inequality

$$\tfrac{1}{2}\,\text{erfc}(x) < \text{e}^{-x^2} \tag{5}$$

which yields

$$P_b(e) < \sum_{w=1}^{k} \frac{w}{k}\,[A(w, H)]|_{H=\text{e}^{-R_cE_b/N_0}}. \tag{6}$$

From (3) and (6), we conclude that, in order to upper-bound the bit error probability for any linear block code, we need to evaluate its CWEF. As a consequence, also for concatenated codes with interleavers we can use (3) and (6), provided that we are able to compute the CWEF of the overall code assuming that the CWEF's of the constituent codes (CC's) are known. This has been done already for "turbo codes," i.e., parallel concatenated codes, in [8]. In the following, we will show how to extend those results to the case of serial concatenation.

[1] It must be noted that E_b/N_0 is *not* the signal-to-noise ratio that can be measured in the channel, which, indeed, is lower by the factor R_c.

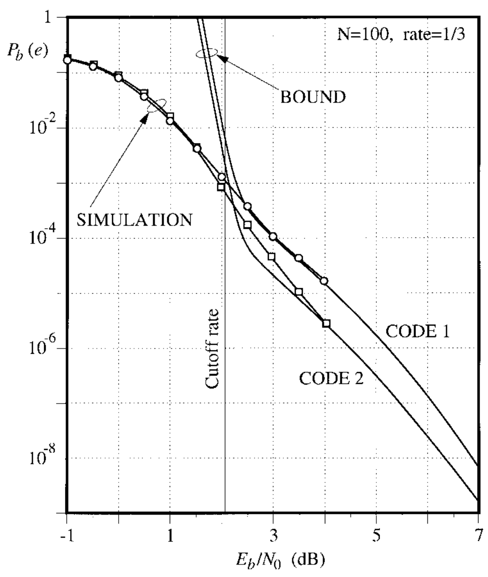

Fig. 1. Comparison of bit error probability curves obtained through union bounds and simulations for two parallel concatenated convolutional codes over an additive Gaussian noise channel. Also indicated is the E_b/N_0 value corresponding to the channel cutoff rate.

Before starting the analysis leading to the evaluation of the CWEF of a serially concatenated code (SCC) with interleaver, a warning to the readers is necessary. Both (3) and (6) stem from the union bound, stating that the probability of a union of events is less than or equal to the sum of the probabilities of the individual events. The union bound is used extensively as an upper limit to the error probabilities for digital transmission systems. The sums of the individual probabilities in the right-hand sides of (3) and (6), however, are not probabilities themselves, and can thus assume large values much greater than one. In fact, it is common knowledge in the field that union bounds are very close to the true probability in the case of maximum-likelihood decoding for medium-high signal-to-noise ratios, whereas they tend to diverge for low signal-to-noise ratios. A widely accepted rule of thumb is that the signal-to-noise ratio where they start to become unreliable is the one yielding the cutoff rate of the channel. The behavior of the bounds is illustrated as a typical example in Fig. 1, where we plot the bounds for two different rate $1/3$ parallel concatenated codes and compare them with simulation results obtained using the suboptimum, iterative decoding algorithm proposed to decode turbo codes. Also drawn in the figure is the E_b/N_0 corresponding to the channel cutoff rate.

Some general comments, partly based on Fig. 1, are appropriate:

- As previously anticipated, the upper bounds based on the union bound diverge at a signal-to-noise ratio close to the channel cutoff rate. Obtaining tighter upper bounds capable of extending the validity interval of the union bounds for concatenated codes is an important, and still widely open, topic for research. The new bounds could be based on the technique successfully employed in [9] for convolutional codes, or on the classical Gallager bound [10]. A successful application of the Gallager bound to parallel concatenated codes with interleavers has been described in [11], where it is shown that the new bound extends the validity of the union bound for some range of signal-to-noise ratios below the channel cutoff rate, typically 0.5 dB. On the other hand, those attempts would still be based on the hypothesis of maximum-likelihood decoding. Thinking of applying them to the suboptimum iterative decoding seems not realistic.

- To obtain the divergence of the union bound one needs to compute a very large number of terms for the summation in the right-hand side of (3), or (6), and this was indeed the case for the example to which the curves of Fig. 1 refer. In that case, however, the interleaver length N was limited to 100. When N becomes very large, as it is required to approach the channel capacity, only a limited number of terms in the summations (3) and (6) can be obtained with a reasonable computational complexity. As a consequence, the obtained upper bounds are still very accurate above the channel cutoff rate, but may not present the divergence at cutoff rate. In those cases, the reader should only consider as reliable the bit error probability values above the cutoff rate, or perhaps half a decibel below it, according to the results of [11]. A way to overcome this drawback, and indeed to always show the divergence of the union bound, has been proposed in [12]. It has, however, to rely on the looser bound (6).

- The main tool used in this paper to analytically predict the performance and to find design rules about the main CC's parameters is a union bound. We have seen, on the other hand, that the union bound is tight only for medium-high signal-to-noise ratios. One could then question the validity of the approach, which suffers the paradox of using a bound not applicable to very low signal-to-noise ratios in order to design coding schemes intended to work near channel capacity.

  We are conscious of this inconsistency, yet, for one hand, have simply nothing better to propose, and, on the other hand, we had widely verified by simulation that the parallel concatenated codes designed on the basis of our rules are indeed very good also at very low signal-to-noise ratios (see [13] and [14]). In the remainder of this paper, we will show that this heuristic validation of the design rules also holds for serially concatenated codes with interleavers.

- The last observation concerns still another inconsistency, resulting from the fact that we are using bounds based on maximum-likelihood decoding to design codes that are decoded according to a different, suboptimum algorithm. Also in this case we invoke the heuristic validation stemming from a large number of simulations, which show the convergence of the simulated performance toward the analytical bounds. In Fig. 1, where the simulated points have been obtained with the suboptimum, iterative decoding algorithm, we see a nice example of this behavior.

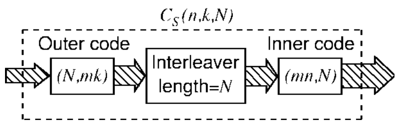

Fig. 2. Serially concatenated $(n, k, N = mp)$ block code.

B. Evaluating the Bit Error Probability Upper Bound for Serially Concatenated Block Codes

We will now show how to apply the union bounds (3) and (6) to the case of serially concatenated codes with interleavers. For simplicity of the presentation, we begin considering serially concatenated block codes (SCBC's).

The scheme of two serially concatenated block codes is shown in Fig. 2. It is composed of two cascaded CC's, the *outer* (p, k) code C_o with rate $R_c^o = k/p$ and the *inner* (n, p) code C_i with rate $R_c^i = p/n$, linked by an interleaver of length $N = mp$ that is an integer multiple of the length p of the outer codewords. The scheme works as follows: the mp bits of a number m of codewords of the outer code are written into the interleaver of length $N = mp$, and read in a different order according to the permutation performed by the interleaver. The sequence of N bits at the output of the interleaver is then sent in blocks of length p to the inner encoder.

The overall SCBC is then an (n, k) code with rate

$$R_c^S = R_c^o \times R_c^i = k/n$$

and we will refer to it as the $(n, k, N = mp)$ code C_S.

In the following, we will derive an upper bound to the ML performance of the overall code C_S, assuming first that $m = 1$, and then extending the result to the general case. We assume that the outer and inner CC's are linear, so that also the SCBC is linear and the *uniform error property* applies, i.e., the bit error probability can be evaluated assuming that the all-zero codeword has been transmitted.

In order to apply the upper bounds (3) and (6) to the SCBC, we need to evaluate the CWEF of the code C_S, assuming that we know the CWEF's of the CC's.

If p is low, we can compute the coefficients $A_{w,h}$ of the CWEF $A(w, H)$ (2) by letting each individual information word with weight w be first encoded by the outer encoder C_o and then, after the p bits of the outer codeword have been permuted by the interleaver, be encoded by the inner encoder C_i originating an inner codeword with a certain weight. After repeating this procedure for all the information words with weight w, we should count the inner codewords with weight h, and their number would be the value of $A_{w,h}$. When k is large, or, in the case $N = mp$, when m is large, the previous operation becomes too complex, and we must resort to a different approach.

The key point, here, is that we would like to obtain a simple relationship between the CWEF's of the two CC's, an operation that is prevented by the fact that the knowledge of the information word weight is not enough to obtain the weight of the inner codeword, which, instead, depends on the weight of the outer codeword and on the permutation induced by the interleaver.

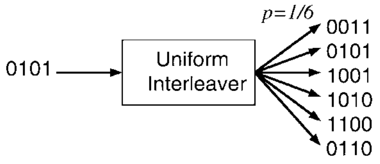

Fig. 3. The action of a uniform interleaver of length 4 on sequences of weight 2.

As in [8] and [15], a crucial step in the analysis consists in replacing the actual interleaver that performs a permutation of the N input bits with an abstract interleaver called *uniform interleaver*, defined as a probabilistic device that maps a given input word of weight l into all distinct $\binom{N}{l}$ permutations of it with equal probability $P = 1/\binom{N}{l}$ (see Fig. 3).

Use of the uniform interleaver permits the computation of the "average" performance of the SCBC, intended as the expectation of the performance of SCBC's using the same CC's, taken over the set of all interleavers of a given length. A theorem proved in [8] guarantees the meaningfulness of the average performance, in the sense that there will always be, for each value of the signal-to-noise ratio, at least one particular interleaver yielding performance better than or equal to that of the uniform interleaver.

Let us define the IOWEF and the CWEF of the SCBC C_S as $A^{C_S}(W, H)$ and $A^{C_S}(w, H)$. Their definition and meaning are the same as in (1) and (2).

As seen, to apply the bounds (3) and (6) to the bit error probability we need to evaluate the CWEF of the SCBC from the knowledge of the CWEF's of the outer and inner codes, which we call $A^{C_o}(w, L)$ and $A^{C_i}(l, H)$, where the first enumerates the weight distributions of the outer codewords generated by information words of weight w, and the second enumerates the weight distributions of the inner codewords generated by outer codewords of weight l.

To do this, we exploit the properties of the uniform interleaver, which transforms a codeword of weight l at the output of the outer encoder into all its distinct $\binom{N}{l}$ permutations. As a consequence, each codeword of the outer code C_o of weight l, through the action of the uniform interleaver, enters the inner encoder generating $\binom{N}{l}$ codewords of the inner code C_i. Thus the number $A_{w,h}^{C_S}$ of codewords of the SCBC of weight h associated with an information word of weight w is given by

$$A_{w,h}^{C_S} = \sum_{l=0}^{N} \frac{A_{w,l}^{C_o} \times A_{l,h}^{C_i}}{\binom{N}{l}}. \tag{7}$$

From (7) we derive the expressions of the CWEF and IOWEF of the SCBC as

$$A^{C_S}(w, H) = \sum_{l=0}^{N} \frac{A_{w,l}^{C_o} \times A^{C_i}(l, H)}{\binom{N}{l}} \tag{8}$$

$$A^{C_S}(W, H) = \sum_{l=0}^{N} \frac{A^{C_o}(W, l) \times A^{C_i}(l, H)}{\binom{N}{l}} \tag{9}$$

where $A^{C_o}(W, l)$ enumerates the weight distributions of the information words that generate codewords of the outer code with a given weight l.

Example 1: Consider the $(7,3)$ serially concatenated block code obtained by concatenating the $(4,3)$ parity-check code to a $(7,4)$ Hamming code through an interleaver of length $N = 4$. The IOWEF $A^{C_o}(W, L)$ and $A^{C_i}(L, H)$ of the outer and inner code are

$$\begin{aligned}A^{C_o}(W, L) &= 1 + W(3L^2) + W^2(3L^2) + W^3(L^4)\\ A^{C_i}(L, H) &= 1 + L(3H^3 + H^4) + L^2(3H^3 + 3H^4)\\ &\quad + L^3(H^3 + 3H^4) + L^4H^7\end{aligned}$$

so that

$$\begin{array}{l|l} A^{C_o}(W, 0) = 1 & A^{C_i}(0, H) = 1\\ A^{C_o}(W, 1) = 0 & A^{C_i}(1, H) = 3H^3 + H^4\\ A^{C_o}(W, 2) = 3W + 3W^2 & A^{C_i}(2, H) = 3H^3 + 3H^4\\ A^{C_o}(W, 3) = 0 & A^{C_i}(3, H) = H^3 + 3H^4\\ A^{C_o}(W, 4) = W^3 & A^{C_i}(4, H) = H^7 \end{array}$$

Through (9) we then obtain

$$\begin{aligned}A^{C_S}(W, H) &= \sum_{l=0}^{4} \frac{A^{C_o}(W, l) \times A^{C_i}(l, H)}{\binom{N}{l}}\\ &= \frac{1 \cdot 1}{1} + \frac{0 \cdot (3H^3 + H^4)}{4}\\ &\quad + \frac{(3W + 3W^2) \cdot (3H^3 + 3H^4)}{6}\\ &\quad + \frac{0 \cdot (H^3 + 3H^4)}{4} + \frac{W^3 \cdot H^7}{1}\\ &= 1 + W(1.5H^3 + 1.5H^4)\\ &\quad + W^2(1.5H^3 + 1.5H^4) + W^3H^7. \qquad \square\end{aligned}$$

Previous results (8) and (9) can be easily generalized to the more interesting case of an interleaver with length N being an integer multiple (by a factor $m > 1$) of the length of the outer codewords. Denoting by $A^{C_o^m}(W, L)$ the IOWEF of the new (mp, mk) outer code, and similarly by $A^{C_i^m}(L, H)$ the IOWEF of the new (mn, mp) inner code, it is straightforward to obtain

$$\begin{aligned}A^{C_o^m}(W, L) &= [A^{C_o}(W, L)]^m\\ A^{C_i^m}(L, H) &= [A^{C_i}(L, H)]^m. \end{aligned} \tag{10}$$

From the IOWEF's (7)–(10), we obtain the CWEF's $A^{C_o^m}(W, l)$ and $A^{C_i^m}(l, H)$ of the new CC's, and, finally, through (8) and (9), the CWEF and IOWEF of the new $(n, k, N = mp)$ SCBC C_S^m

$$A^{C_S^m}(w, H) = \sum_{l=0}^{N} \frac{A^{C_o^m}_{w,l} \times A^{C_i^m}(l, H)}{\binom{N}{l}} \tag{11}$$

$$A^{C_S^m}(W, H) = \sum_{l=0}^{N} \frac{A^{C_o^m}(W, l) \times A^{C_i^m}(l, H)}{\binom{N}{l}}. \tag{12}$$

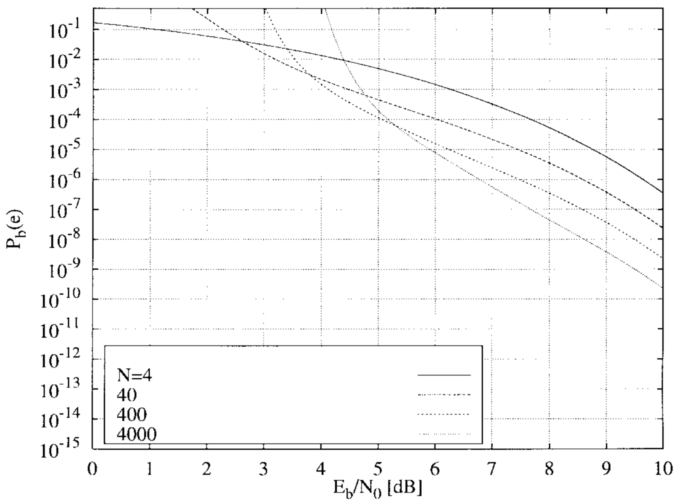

Fig. 4. Analytical bounds for serially concatenated block code of Example 2 (SCBC1 in Table I).

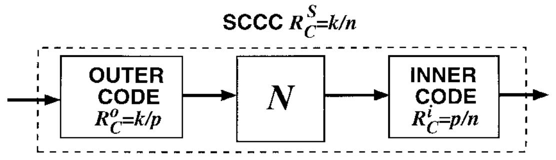

Fig. 5. Serially concatenated (n, k, N) convolutional code.

Example 2: Consider again the CC's of Example 1, linked by an interleaver of length $N = 4m$, and use (2), (3), and (11). The so obtained upper bound to the bit error probability is plotted in Fig. 4 for various values of the integer m. The curves show the *interleaver gain*, defined as the factor by which the bit error probability is decreased with the interleaver length at a given signal-to-noise ratio. Contrary to parallel concatenated block codes [8], the curves do not exhibit the interleaver gain saturation. Rather, the bit error probability seems to decrease regularly with m as m^{-1}. We will explain this behavior in Section III. $\square$

C. Serially Concatenated Convolutional Codes

The structure of a serially concatenated convolutional code is shown in Fig. 5. It refers to the case of two convolutional CC's, the outer code C_o with rate $R_c^o = k/p$, and the inner code C_i with rate $R_c^i = p/n$, joined by an interleaver of length N bits, generating an SCCC C_S with rate $R_c^S = k/n$. N will be assumed to be an integer multiple[2] of p. We assume, as before, that the convolutional CC's are linear, so that the SCCC is linear as well, and the uniform error property applies.

The exact analysis of this scheme can be performed by appropriate modifications of that described in [8] for PCCC's. It requires the use of a *hypertrellis* having as *hyperstates* pairs of states of outer and inner codes. The hyperstates S_{ij} and S_{lm} are joined by a *hyperbranch* that consists of all pairs of paths

[2] Actually, this constraint is not necessary. We can choose in fact inner and outer codes of any rates $R_c^i = k_i/n_i$ and $R_c^o = k_o/n_o$, constraining the interleaver to be an integer multiple of the minimum common multiple of n_o and k_i, i.e., $N = K \cdot \text{mcm}(n_o, k_i)$. This generalization, though, leads to more complicated expressions and is not considered in the following.

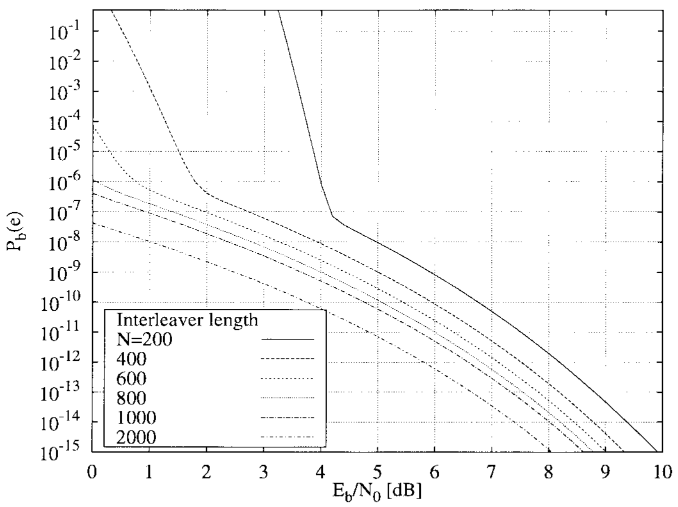

Fig. 6. Analytical bounds for the serially concatenated convolutional code of Example 3 (SCCC1 of Table II).

with length N/p that join states s_i, s_l of the inner code and states s_j, s_m of the outer code, respectively. Each hyperbranch is thus an equivalent SCBC labeled with an IOWEF that can be evaluated as explained in the previous subsection. From the hypertrellis, the upper bound to the bit error probability can be obtained through the standard transfer function technique employed for convolutional codes [10]. As proved in [8], when the length of the interleaver is significantly greater than the constraint lengths of the CC's, an accurate approximation of the exact upper bound consists in retaining only the branch of the hypertrellis joining the hyperstates S_{00}, S_{00}. In the following, we will always use this approximation.

Example 3: Consider a rate $1/3$ SCCC formed by an outer 4-state convolutional code with rate $1/2$ and an inner 4-state convolutional code with rate $2/3$, joined by a uniform interleaver of length $N = 200, 400, 600, 800, 1000$, and 2000 (SCCC1 of Table II).

Both encoders are systematic and recursive and the generator matrices are reported in Table III, first and third rows. Using the previously outlined analysis, we have obtained the bit error probability curves shown in Fig. 6. The performance shows a very significant interleaver gain, i.e., lower values of the bit error probability for higher values of N. The interleaver gain seems to behave as N^{-3}. This behavior will be explained in the next section. □

III. Design of Serially Concatenated Codes

In the previous section, we have presented an analytical bounding technique to find the ML performance of SCBC's and SCCC's. For practical applications, SCCC's are to be preferred to SCBC's. One reason is that *a posteriori*-probability algorithms are less complex for convolutional than for block codes, another is that the interleaver gain can be greater for convolutional CC's, provided that they are suitably designed. Hence, we deal mainly with the design of SCCC's, extending our conclusions to SCBC's when appropriate.

Before delving deeply into the analytical derivation, it is important to say a few words about the design methodology. The performance of a concatenated code with interleaver, for both cases of parallel and serial concatenation, depends on the constituent codes and on the interleaver in a strictly interdependent manner. The joint design of CC's and the interleaver, however, is a hopeless goal, and the only way to achieve significantly good results seems to pass through a *decoupled* design, in which one first designs the CC's, and then tailors the interleaver on their characteristics. Our approach to achieve this goal resides once more in the use of the uniform interleaver, which yields to an average optimization of the CC's, followed by a customization of the actual interleaver. Only the first step, i.e., the design of CC's, will be treated here.

Consider the SCCC depicted in Fig. 5. Its performance can be approximated by that of the equivalent block code whose IOWEF labels the branch of the hypertrellis joining the zero states of outer- and inner-code trellises. Denoting by $A^{C_S}(w, H)$ the CWEF of this equivalent block code, we can rewrite the upper bound (6) as[3]

$$\begin{aligned} P_b(e) &< \sum_{w=w_m^o}^{NR_c^o} \frac{w}{NR_c^o} A^{C_S}(w, H)\big|_{H=e^{-R_c E_b/N_0}} \\ &= \sum_{h=h_m}^{N/R_c^i} \sum_{w=w_m^o}^{NR_c^o} \frac{w}{NR_c^o} A_{w,h}^{C_S} e^{-hR_c E_b/N_0} \end{aligned} \tag{13}$$

where w_m^o is the minimum weight of an input sequence generating an error event of the outer code, and h_m is the minimum weight[4] of the codewords of C_S. By error event of a convolutional code, we mean a sequence diverging from the zero state at time zero and remerging into the zero state at some discrete time $j > 0$. For constituent block codes, an error event is simply a codeword.

The coefficients $A_{w,h}^{C_S}$ of the equivalent block code can be obtained from (7), once the quantities $A_{w,l}^{C_o}$ and $A_{l,h}^{C_i}$ of the CC's are known. To evaluate them, consider a rate $R = p/n$ convolutional code C with memory[5] 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. Let

$$A(l, H, j) = \sum_h A_{l,h,j} H^h \tag{14}$$

be the weight enumerating function of sequences of the convolutional code that concatenate j error events with total input information weight l. The coefficient $A_{l,h,j}$ represents the number of sequences of weight h, input weight l, and number of concatenated error events j. Its meaning is pictorially clarified in Fig. 7, where it can be noticed that, by "concatenated," we mean actually that the j error events are

[3] In the following, a subscript "m" will denote "minimum," and a subscript "M" will denote "maximum."

[4] Since the input sequences of the inner code are *not* unconstrained independent and identically distributed (i.i.d.) binary sequences, but, instead, codewords of the outer code, h_m can be greater than the inner code free distance d_f^i.

[5] By memory, we mean the maximum length of the shift registers contained in the encoder.

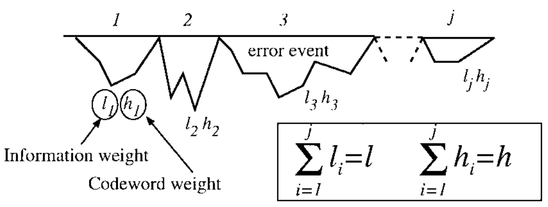

Fig. 7. The meaning of the coefficients $A_{l,h,j}$.

adjacent, in the sense that each one starts immediately where the previous ends, without any gap in between.

For N much larger than the memory of the convolutional code, the coefficient $A^C_{l,h}$ of the CWEF of the equivalent block code can be upper-bounded by[6]

$$A^C_{l,h} \le \sum_{j=1}^{n_M} \binom{N/p}{j} A_{l,h,j} \tag{15}$$

where n_M, the largest number of error events concatenated in a codeword of weight h and generated by a weight l information sequence, is a function of h and l that depends on the encoder, as we will see later.

Let us return now to the block code equivalent to the SCCC. Using previous result (15) with $j = n^o$ and $j = n^i$ for the outer and inner code, respectively,[7] we can write

$$A^{C_o}_{w,l} \le \sum_{n^o=1}^{n^o_M} \binom{N/p}{n^o} A^o_{w,l,n^o}$$

and

$$A^{C_i}_{l,h} \le \sum_{n^i=1}^{n^i_M} \binom{N/p}{n^i} A^i_{l,h,n^i}.$$

Substituting them into (7), we obtain an upper bound to the value of the coefficients $A^{C_S}_{w,h}$ of the serially concatenated block code equivalent to the SCCC in the form

$$A^{C_S}_{w,h} \le \sum_{l=d^o_f}^{N} \sum_{n^o=1}^{n^o_M} \sum_{n^i=1}^{n^i_M} \frac{\binom{N/p}{n^o}\binom{N/p}{n^i}}{\binom{N}{l}} A^o_{w,l,n^o} A^i_{l,h,n^i} \tag{16}$$

where d^o_f is the free distance of the outer code. By free distance d_f we mean the minimum Hamming weight of error events for convolutional CC's, and the minimum Hamming weight of codewords for block CC's.

To proceed further, we need to replace the three binomial coefficients in (16). In order to obtain an upper bound to the right-hand side of (16), we will replace the two binomials in the numerator using the upper bound

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

and the binomial in the denominator using the lower bound

$$\binom{N}{l} > \frac{(N-l+1)^l}{l!} > \frac{N^l}{l^l l!}.$$

Notice that the bounds are tight for large N and for $n, l \ll N$, which will be seen to always happen in our case.

Substitution of these bounds in (16) yields

$$A^{C_S}_{w,h} \le \sum_{l=d^o_f}^{N} \sum_{n^o=1}^{n^o_M} \sum_{n^i=1}^{n^i_M} N^{n^o+n^i-l} \cdot \frac{l^l l!}{p^{n^o+n^i} n^o! n^i!} A^o_{w,l,n^o} A^i_{l,h,n^i}. \tag{17}$$

Finally, substituting (17) into (13), gives the bit error probability bound in the form

$$P_b(e) \le \sum_{h=h_m}^{N/R^i_c} e^{-hR_cE_b/N_0} \sum_{w=w^o_m}^{NR^o_c} \sum_{l=d^o_f}^{N} \sum_{n^o=1}^{n^o_M} \sum_{n^i=1}^{n^i_M} N^{n^o+n^i-l-1} \cdot \frac{l^l l!}{p^{n^o+n^i-1} n^o! n^i!} \frac{w}{k} A^o_{w,l,n^o} A^i_{l,h,n^i}. \tag{18}$$

Using (18) as the starting point, we will obtain some important design considerations. The bound (18) to the bit error probability is obtained by adding terms of the first summation with respect to the SCCC weights h. The coefficients of the exponentials in h depend, among other parameters, on N. For large N, and for a given h, the dominant coefficient of the exponentials in h is the one for which the exponent of N is maximum. Define this maximum exponent as

$$\alpha(h) \triangleq \max_{w,l}\{n^o + n^i - l - 1\}. \tag{19}$$

Evaluating $\alpha(h)$ in general is not possible without specifying the CC's. 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 SCC is dominated by the first term of the summation with respect to h, corresponding to the minimum value $h = h_m$. Remembering that by definition, n^i_M and n^o_M are the maximum number of concatenated error events in codewords of the inner and outer code of weights h_m and l, respectively, the following inequalities hold true:

$$n^i_M \le \left\lfloor \frac{h_m}{d^i_f} \right\rfloor \tag{20}$$

$$n^o_M \le \left\lfloor \frac{l}{d^o_f} \right\rfloor \tag{21}$$

and

$$\alpha(h_m) \le \max_l \left\{ \left\lfloor \frac{h_m}{d^i_f} \right\rfloor + \left\lfloor \frac{l}{d^o_f} \right\rfloor - l - 1 \right\} = \left\lfloor \frac{h_m}{d^i_f} \right\rfloor + \left\lfloor \frac{l_m(h_m)}{d^o_f} \right\rfloor - l_m(h_m) - 1 \tag{22}$$

[6]The upper bound is obtained neglecting the length of error events compared to N, and assuming 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 words or, equivalently, trellis steps.

[7]In the following, superscripts "o" and "i" will refer to quantities pertaining to outer and inner code, respectively.

where $l_m(h_m)$ is the minimum weight l of codewords of the outer code yielding a codeword of weight h_m of the inner code, and $\lfloor x \rfloor$ means "integer part of x."

In most cases,[8] $l_m(h_m) < 2d_f^o$, and $h_m < 2d_f^i$, so that $n_M^i = n_M^o = 1$, and (22) becomes

$$\alpha(h_m) = 1 - l_m(h_m) \leq 1 - d_f^o. \tag{23}$$

The result (23) shows that the exponent of N corresponding to the minimum weight of SCCC codewords is always negative for $d_f^o \geq 2$, thus yielding an interleaver gain at high E_b/N_0. Substitution of the exponent $\alpha(h_m)$ into (18) truncated to the first term of the summation with respect to h yields the following result, asymptotic with respect to E_b/N_0:

$$P_b(e) \overset{\sim}{\leq} B_m N^{1-d_f^o} \exp(-h_m R_c E_b/N_0) \tag{24}$$

where the constant B_m is

$$B_m = \frac{A^i_{l_m(h_m), h_m, 1} [l_m(h_m)]! l_m^{l_m}}{kp} \sum_{w \in \mathcal{W}_m} w A^o_{w, l_m(h_m), 1}$$

and $\mathcal{W}_m$ is the set of input weights w that generate codewords of the outer code with weight $l_m(h_m)$.

Expression (24) suggests the following conclusions.

- For the values of E_b/N_0 and N where the SCCC performance is dominated by its free distance $d_f^{C_S} = h_m$, increasing the interleaver length yields a gain in performance.
- To increase the interleaver gain, one should choose an outer code with large d_f^o.
- To improve the performance with E_b/N_0, one should choose an inner- and outer-code combination such that h_m is large.

These conclusions do not depend on the structure of the CC's, and thus they yield for both recursive and nonrecursive encoder.

The curves of Fig. 4 showing the performance of the various SCBC's of Example 1 with increasing interleaver length, however, also show a different phenomenon: for a given E_b/N_0, there is a value of N above which the bound diverges. In other words, there seem to be coefficients of the exponents in h, for $h > h_m$, that increase with N.

To investigate this phenomenon, we will evaluate the largest exponent of N, defined as

$$\alpha_M \triangleq \max_h \{\alpha(h)\} = \max_{w, l, h} \{n^o + n^i - l - 1\}. \tag{25}$$

This exponent will permit 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. As we will see, nonrecursive encoders and block encoders show the same behavior.

[8]This will be seen in the examples that follow and corresponds to the most favorable situation.

1) Block and Nonrecursive Convolutional Inner Encoders: Consider the inner encoder and its impact on the exponent of N in (25). For a nonrecursive inner encoder, we have $n_M^i = l$. In fact, every input sequence with weight one generates a finite-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. Since the uniform interleaver generates all possible permutations of its input sequences, this event will certainly occur.

Thus from (25) we have

$$\alpha_M = n_M^o - 1 \geq 0$$

and interleaving gain is not allowed. This conclusion holds true for both SCCC's employing nonrecursive inner encoders and for all SCBC's, since block codes have codewords corresponding to input words with weight equal to one.

For those SCC's we *always* have, for some h, coefficients of the exponential in h of (18) that increase with N, and this explains the divergence of the bound arising, for each E_b/N_0, when the coefficients increasing with N become dominant.

2) Recursive Inner Encoders: For recursive convolutional encoders, the minimum weight of input sequences generating error events is 2 (see [13] and [16]). As a consequence, an input sequence of weight l can generate at most $\lfloor l/2 \rfloor$ error events.

Assuming that the inner encoder of the SCCC is recursive, the maximum exponent of N in (25) becomes

$$\begin{aligned}\alpha_M &= \max_{w,l} \left\{ n_M^o + \left\lfloor \frac{l}{2} \right\rfloor - l - 1 \right\} \\ &= \max_{w,l} \left\{ n_M^o - \left\lfloor \frac{l+1}{2} \right\rfloor - 1 \right\}. \end{aligned} \tag{26}$$

The maximization involves l and w, since n_M^o depends on both quantities. In fact, remembering its definition as the maximum number of concatenated error events of codewords of the outer code with weight l generated by input words of weight w, it is straightforward to obtain

$$n_M^o = \min \left\{ \left\lfloor \frac{w}{w_m} \right\rfloor, \left\lfloor \frac{l}{d_f^o} \right\rfloor \right\} \leq \left\lfloor \frac{l}{d_f^o} \right\rfloor. \tag{27}$$

Substituting now the last inequality (27) into (26) yields

$$\alpha_M \leq \max_l \left\{ \left\lfloor \frac{l}{d_f^o} \right\rfloor - \left\lfloor \frac{l+1}{2} \right\rfloor - 1 \right\}. \tag{28}$$

To perform the maximization of the right-hand side of (28), consider first the case

$$l = q d_f^o, \qquad q \text{ integer}$$

so that

$$\alpha_M \leq \max_q \left\{ q - \left\lfloor \frac{q d_f^o + 1}{2} \right\rfloor - 1 \right\}. \tag{29}$$

The right-hand sideof (29) is maximized, for $d_f^o \geq 2$, by choosing $q = 1$. On the other hand, for

$$q d_f^o \leq l < (q+1) d_f^o$$

the most favorable case is $l = qd_f^o$, which leads us again to the previously discussed situation. Thus the maximization requires $l = d_f^o$. For this value, on the other hand, we have from (27) $n_M^o \leq 1$, and the inequality becomes an equality if $w \in \mathcal{W}_f$, where $\mathcal{W}_f$ is the set of input weights w that generate codewords of the outer code with weight $l = d_f^o$. In conclusion, the largest exponent of N is given by

$$\alpha_M = -\left\lfloor \frac{d_f^o + 1}{2} \right\rfloor. \tag{30}$$

The value (30) of α_M shows that the exponents of N in (18) are always negative integers. Thus for all h, the coefficients of the exponents in h decrease with N, and we always have an interleaver gain.

Denoting by $d_{\text{f, eff}}^i$, as in [8], the minimum weight of codewords of the inner code generated by weight-2 input sequences, we obtain a different weight $h(\alpha_M)$ for even and odd values of d_f^o.

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

$$h(\alpha_M) = \frac{d_f^o d_{\text{f, eff}}^i}{2}$$

since it is the weight of an inner codeword that concatenates $d_f^o/2$ error events with weight $d_{\text{f, eff}}^i$.

Substituting the exponent α_M into (18), approximated only by the term of the summation with respect to h corresponding to $h = h(\alpha_M)$, yields the following result, asymptotic with respect to E_b/N_0:

$$P_b(e) \overset{\sim}{\leq} B_{\text{even}} N^{-d_f^o/2} \exp\left[-\frac{d_f^o d_{\text{f, eff}}^i}{2} R_c E_b/N_0\right] \tag{31}$$

where

$$\begin{aligned} B_{\text{even}} &= \frac{A^i_{d_f^o,\, h(\alpha_M),\, (d_f^o/2)} (d_f^o)^{d_f^o} d_f^o!}{k p^{d_f^o/2} (d_f^o/2)!} \sum_{w \in \mathcal{W}_f} w A^o_{w,\, d_f^o,\, 1} \\ &\leq A^i_{d_f^o,\, h(\alpha_M),\, (d_f^o/2)} w_{M,\,f} N_f^o \frac{(d_f^o)^{d_f^o} d_f^o!}{k p^{d_f^o/2} (d_f^o/2)!}. \end{aligned} \tag{32}$$

In (32), $w_{M,\,f}$ is the maximum input weight yielding outer codewords with weight equal to d_f^o, and N_f^o is the number of such codewords.

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

$$h(\alpha_M) = \frac{(d_f^o - 3) d_{\text{f, eff}}^i}{2} + h_m^{(3)} \tag{33}$$

where $h_m^{(3)}$ is the minimum weight of sequences of the inner code generated by a weight–3 input sequence. In this case, in fact, we have

$$n_M^i = \frac{d_f^o - 1}{2}$$

concatenated error events, of which $n_M^i - 1$ generated by weight-2 input sequences and one generated by a weight-3 input sequence.

Thus substituting the exponent α_M into (18) approximated by keeping only the term of the summation with respect to h corresponding to $h = h(\alpha_M)$ yields the following result, asymptotic with respect to E_b/N_0:

$$\begin{aligned} P_b(e) &\overset{\sim}{\leq} B_{\text{odd}} N^{-\frac{d_f^o+1}{2}} \\ &\cdot \exp\left\{-\left[\frac{(d_f^o - 3) d_{\text{f, eff}}^i}{2} + h_m^{(3)}\right] R_c E_b/N_0\right\} \end{aligned} \tag{34}$$

where

$$\begin{aligned} B_{\text{odd}} &= \frac{A_{d_f^o,\, h(\alpha_M),\, (d_f^o-1)/2} (d_f^o)^{d_f^o} d_f^o!}{k p^{(d_f^o-1)/2} [(d_f^o-3)/2]!} \sum_{w \in \mathcal{W}_f} w A^o_{w,\, d_f^o,\, 1} \\ &\leq A_{d_f^o,\, h(\alpha_M),\, (d_f^o-1)/2} w_{M,\,f} N_f^o \frac{(d_f^o)^{d_f^o} d_f^o!}{k p^{(d_f^o-1)/2} [(d_f^o-3)/2]!}. \end{aligned} \tag{35}$$

In both cases of d_f^o even and odd, we can draw from (31) and (34) a few important design considerations:

- in contrast with the case of block codes and nonrecursive convolutional inner encoders, the use of a recursive convolutional inner encoder always yields an interleaver gain. As a consequence, the first design rule states that *the inner encoder must be a convolutional recursive encoder*.
- The coefficient $h(\alpha_M)$, which multiplies the signal-to-noise ratio E_b/N_0 in (18), increases for increasing values of $d_{\text{f, eff}}^i$. Thus we deduce that *the effective free distance of the inner encoder must be maximized*. Both this and the previous design rule had been stated also for PCCC's[9] [13]. As a consequence, the recursive convolutional encoders optimized for use in PCCC's (see tables in [13] and [17]) can be employed altogether as inner CC in SCCC's.

  From (33), however, we can infer that also the parameter $h_m^{(3)}$, namely the minimum weight of inner code sequences generated by weight-3 input sequences, is important for odd values of d_f^o. This may lead to a choice for the inner code that is in partial disagreement with the design rules obtained for PCCC's in [13]. Those design rules stated that optimum constituent encoders for PCCC's should have a feedback connection made according to a primitive polynomial. For SCCC's with odd d_f^o, it is often convenient to choose an inner encoder whose feedback polynomial contains $(1 + D)$ as a factor; this choice, in fact, provided that the feedforward polynomial does not cancel such factor, eliminates all error events of the inner code originated by odd-weight input sequences, thus yielding $h_m^{(3)} = \infty$. As a consequence, the parameter $h(\alpha_M)$ would not be obtained from (33) anymore, but, instead, from an input sequence to the inner code with weight larger than d_f^o, yielding a greater value for $h(\alpha_M)$. In Section III, we will see an example of code designed according to that procedure.
- The interleaver gain is equal to $N^{-d_f^o/2}$ for even values of d_f^o and to $N^{-(d_f^o+1)/2}$ for odd values of d_f^o. As a consequence, *we should choose*, compatibly with the desired rate R_c of the SCCC, *an outer code with a large and, possibly, odd value of the free distance*.

[9] For PCCC's, however, both CC's had to comply with those design rules.

TABLE I
DESIGN PARAMETERS OF CONSTITUENT CODES AND SERIALLY CONCATENATED BLOCK CODES FOR THREE SERIALLY CONCATENATED BLOCK CODES

Code	Outer code			Inner code				SCBC	
	Code type	w_m^o	d_f^o	Code type	w_m^i	d_f^i	$d_{f,\mathrm{eff}}^i$	h_m	$\alpha(h_m)$
SCBC1	Parity check (4,3)	1	2	Hamming (7,4)	1	3	3	3	-1
SCBC2	Parity check (5,4)	1	2	BCH (15,5)	1	7	7	7	-1
SCBC3	Hamming (7,4)	1	3	BCH (15,7)	1	5	5	5	-2

- As to other outer code parameters, N_f^o and $w_{M,f}$ should be minimized. In other words, we should have the minimum number of input sequences generating free distance error events of the outer code, and their input weights should be minimized. Since nonrecursive encoders have error events with $w = 1$, and, in general, less input errors associated with error events at free distance [18], it can be convenient *to choose as outer code a nonrecursive encoder* with minimum N_f^o and $w_{M,f}$. Conventional nonrecursive convolutional codes found in books (see, for example, [7]) are appropriate.

C. Examples Confirming the Design Rules

To confirm the design rules obtained asymptotically, i.e., for large signal-to-noise ratio and large interleaver lengths N, we evaluate the upper bound (3) to the bit error probability for several block and convolutional SCC's, with different interleaver lengths, and compare their performance with those predicted by the design guidelines.

1) Serially Concatenated Block Codes: We consider three different SCBC's obtained as follows: the first is the $(7, 3, N = 4m)$ SCBC described in Example 2, the second is a $(15, 4, N = 5m)$ SCBC using as outer code a $(5, 4)$ parity-check code and as inner code a $(15, 5)$ BCH code, and the third is a $(15, 4, N = 7m)$ SCBC using as outer code a $(7, 4)$ Hamming code and as inner code a $(15, 7)$ BCH code. Note that the second and third SCBC's have the same rate, $4/15$.

The outer, inner, and SCBC code parameters introduced in the design analysis of Section III are listed in Table I.

In Figs. 4, 8, and 9, we plot the bit error probability bounds for the SCBC's 1, 2, and 3 of Table I.

Code SCBC1 has $d_f^o = 2$; thus from (23), we expect an interleaver gain going as N^{-1}. This is confirmed by the curves of Fig. 4, which, for a fixed and sufficiently large signal-to-noise ratio, show a decrease in $P_b(e)$ of a factor 10, when N increases from 4 to 40, from 40 to 400, and from 400 to 4000. Moreover, curves of Fig. 4 show a divergence of the bound at higher E_b/N_0 for increasing N. This is due to coefficients of terms with $h > h_m$ in (18) that increase with N, whose influence become more important for larger N.

Code SCBC2 has $d_f^o = 2$; thus from (23), we expect the same interleaver gain as for SCBC1, i.e., N^{-1}. This is confirmed by the curves of Fig. 8, which also show the bound divergence predicted in the analysis of Section II.

Code SCBC3 has $d_f^o = 3$; thus from (23), we expect a larger interleaver gain than for SCBC1 and SCBC2, i.e., N^{-2}. This is confirmed by the curves of Fig. 9, which, for a fixed and sufficiently large signal-to-noise ratio, show a decrease in

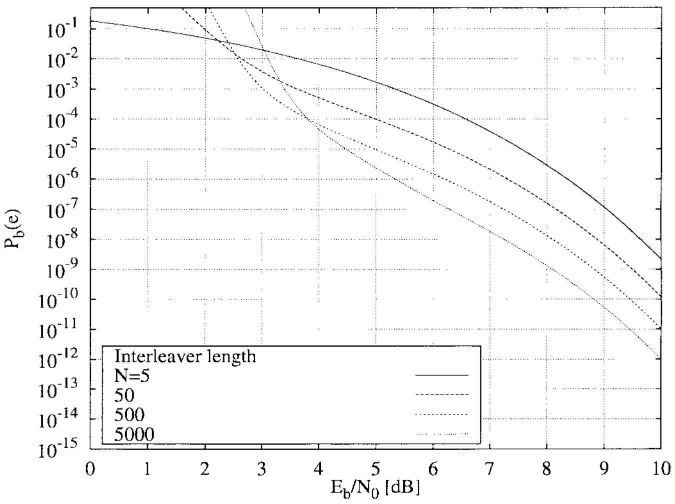

Fig. 8. Analytical bounds for the serially concatenated block code SCBC2.

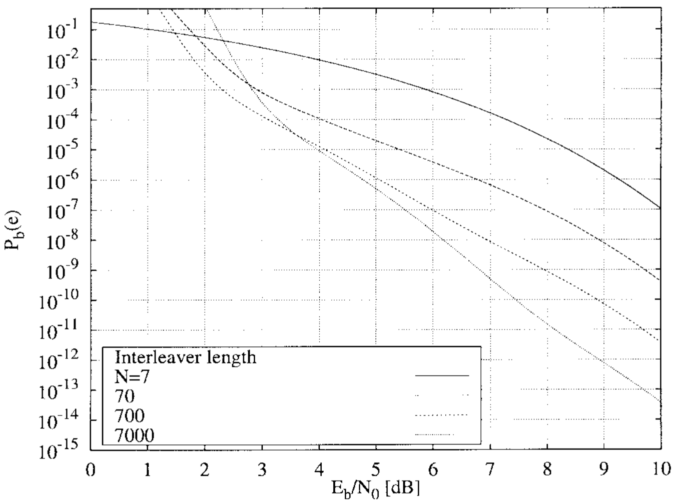

Fig. 9. Analytical bounds for the serially concatenated block code SCBC3.

$P_b(e)$ of a factor 100, when N increases from 7 to 70, from 70 to 700, and from 700 to 7000. As to the bound divergence, we notice a slightly different behavior with respect to previous cases. The curve with $N = 7000$, in fact, denotes a strong influence of coefficients increasing with N for E_b/N_0 lower than 7 dB.

B. Serially Concatenated Convolutional Codes

We consider four different SCCC's obtained as follows: the first, SCCC1, is a $(3, 1, N)$ SCCC (the same of Example 3), using as outer code a 4-state, $(2, 1)$ recursive, systematic

TABLE II
DESIGN PARAMETERS OF CONSTITUENT CODES AND OF SERIALLY CONCATENATED CONVOLUTIONAL CODES FOR FOUR SERIALLY CONCATENATED CONVOLUTIONAL CODES

Code	Outer code			Inner code				SCCC			
	Code	w_m^o	d_f^o	Code	w_m^i	d_f^i	$d_{f,\mathrm{eff}}^i$	h_m	$\alpha(h_m)$	$h(\alpha_M)$	α_M
SCCC1	Rate 1/2 R	2	5	Rate 2/3 R	2	3	4	5	-4	7	-3
SCCC2	Rate 1/2 R	2	5	Rate 2/3 NR	1	3	4	5	-4		
SCCC3	Rate 1/2 NR	1	5	Rate 2/3 R	2	3	4	5	-4	7	-3
SCCC4	Rate 2/3 NR	1	3	Rate 1/2 R	2	5	6	5	-2	5	-2

TABLE III
GENERATING MATRICES FOR THE CONSTITUENT CONVOLUTIONAL CODES

Code description	$G(D)$
Rate 1/2 R	$\begin{bmatrix} 1, & \frac{1+D^2}{1+D+D^2} \end{bmatrix}$
Rate 1/2 NR	$\begin{bmatrix} 1+D+D^2, & 1+D^2 \end{bmatrix}$
Rate 2/3 R	$\begin{bmatrix} 1, & 0, & \frac{1+D^2}{1+D+D^2} \\ 0, & 1, & \frac{1+D}{1+D+D^2} \end{bmatrix}$
Rate 2/3 NR	$\begin{bmatrix} 1+D, & D, & 1 \\ 1+D, & 1, & 1+D \end{bmatrix}$

convolutional encoder and as inner code a 4-state, $(3,2)$ recursive, systematic convolutional encoder. The second, SCCC2, is a $(3,1,N)$ SCCC, using as outer code the same 4-state, $(2,1)$ recursive, systematic convolutional encoder of SCCC1, and as inner code a 4-state, $(3,2)$ nonrecursive convolutional encoder. The third, SCCC3, is a $(3,1,N)$ SCCC using as outer code a 4-state, $(2,1)$ nonrecursive, convolutional encoder, and as inner code the same 4-state, $(3,2)$ recursive, systematic convolutional encoder of SCCC1. The fourth, SCCC4, finally, is a $(6,2,N)$ SCCC, using as outer code a 4-state, $(3,2)$ nonrecursive convolutional encoder, and as inner code a 4-state, $(6,3)$ recursive, systematic convolutional encoder, obtained by concatenating three equal 4-state $(2,1)$ recursive, systematic convolutional encoders.

The outer, inner, and SCCC code parameters introduced in the design analysis in Section III are listed in Table II. In this table, the CC's are identified through the description of Table III.

In Figs. 6, 10, 11, and 12, we plot the bit error probability bounds for the SCCC's 1, 2, 3, and 4 of Table II, with interleaver lengths $N = 200, 400, 600, 800, 1000, 2000$. In Fig. 12, which refers to the code SCCC4 of Table II, the values of the interleaver length are different, i.e., $N = 150, 300, 450, 600, 750, 1500$. This is due to the fact that the outer code for SCCC4 has a rate $2/3$, instead of $1/2$, and thus the interleaver length must be changed in order to guarantee the same input delay for all SCCC's.

Consider first the SCCC's employing as inner CC's recursive, convolutional encoders as suggested in Section III. They are SCCC1, SCCC3, and SCCC4.

Code SCCC1 has $d_f^o = 5$; thus from (34), we expect an interleaver gain behaving as N^{-3}. This is fully confirmed by the curves of Fig. 6, which, for a fixed and sufficiently large signal-to-noise ratio, show a decrease in $P_b(e)$ by a factor 1000, when N increases from 200 to 2000. For an even more accurate confirmation, one can compare the interleaver gain

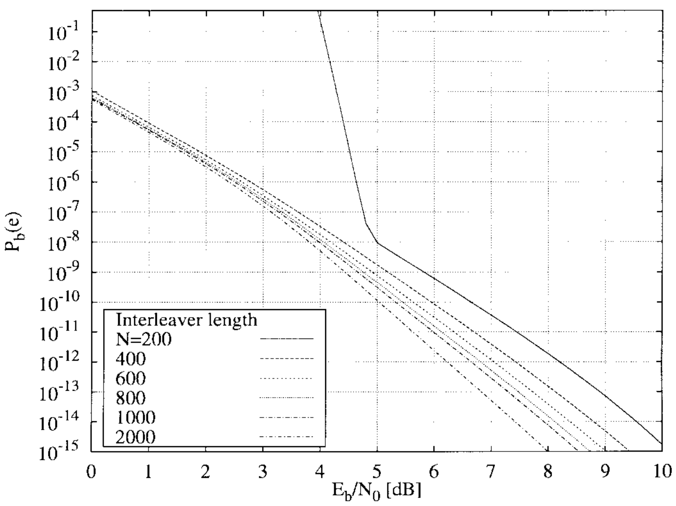

Fig. 10. Analytical bounds for the serially concatenated convolutional code SCCC2.

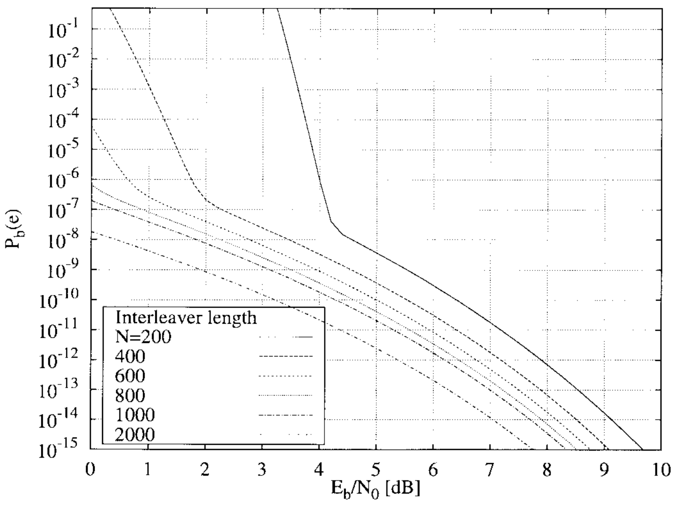

Fig. 11. Analytical bounds for the serially concatenated convolutional code SCCC3.

for every pair of curves in the figure. The curves of Fig. 6 do not show a divergence of the bound at higher E_b/N_0 for increasing N. This is due to the choice of a recursive encoder for the inner code, which guarantees that all coefficients $\alpha(h)$ decrease with N.

Code SCCC3 differs from SCCC1 only in the choice of a nonrecursive outer encoder, which is a four-state encoder (see Table II) with the same d_f^o as for SCCC1, but with $w_m^o = 1$, instead of $w_m^o = 2$. From the design conclusions, we expect a slightly better behavior of this SCCC. This is confirmed by the performance curves of Fig. 11, which present the same interleaver gain as those of SCCC1, but have a slightly lower $P_b(e)$ (the curves for SCCC3 are translated versions of those of SCCC1 by 0.1 dB).

Code SCCC4 employs the same CC's as SCCC2, but reverses their order. It uses as outer code a rate $2/3$ nonrecursive convolutional encoder, and as inner code a rate $1/2$ recursive convolutional encoder. As a consequence, it has a lower $d_f^o = 3$ and a lower $\alpha_M = -2$. Thus from (34), we expect a lower interleaver gain than for SCCC1 and SCCC3, as N^{-2}. This is confirmed by the curves of Fig. 12, which, for a fixed and sufficiently large signal-to-noise ratio, show a decrease in $P_b(e)$ of a factor 100, when N increases from 150 to 1500. On the whole, SCCC4 looses more than 2 dB in coding gain with respect to SCCC3. This result confirms the design rule suggesting the choice of an outer code with as large d_f^o as possible.

Finally, let us consider code SCCC2, which differs from SCCC1 in the choice of a nonrecursive inner encoder, with the same parameters but with the crucial difference of $w_m^i = 1$. Its bit error probability curves are shown in Fig. 10. They confirm the predictions of Section III. We see, in fact, that for low signal-to-noise ratios, say below 4 dB, almost no interleaver gain is obtained. This is because the performance is dominated by the exponent $h(\alpha_M)$, whose coefficient increases with N. On the other hand, for larger signal-to-noise ratios, where the dominant contribution to $P_b(e)$ is the exponent with the lowest value h_m, the interleaver gain makes its appearance. From (24), we foresee a gain as N^{-4}, meaning four orders of magnitude for N increasing from 200 to 2000. Curves in Fig. 10 show a smaller gain (slightly higher than $1/1000$), which is, on the other hand, rapidly increasing with E_b/N_0.

IV. Comparison Between Parallel and Serially Concatenated Codes

In this section, we will use the bit error probability bounds previously derived to compare the performance of parallel ("turbo codes," [8]) and serially concatenated block and convolutional codes.

A. Serially and Parallel Concatenated Block Codes

To obtain a fair comparison, we have chosen the following PCBC and SCBC: the PCBC has parameters $(11, 3, N = 3m)$ and employs two equal $(7, 3)$ systematic cyclic codes with generator $g(D) = (1+D)(1+D+D^3)$; the SCBC, instead, is a $(15m, 4m, N = 7m)$ SCCC, obtained by the concatenation of the $(7, 4)$ Hamming code with a $(15, 7)$ BCH code.

They have almost the same rate ($R_c^S = 0.266$, $R_c^P = 0.273$), and have been compared choosing the interleaver length in such a way that the decoding delay due to the interleaver, measured in terms of input information bits, is the same. Since the interleaver acts on information bits for the PCBC, its length is equal to m times the length of the information word, i.e., $N = 3m$. For the SCBC, instead, the interleaver acts on the codewords of the outer code, and, as a consequence, to yield the same decoding delay measured as number of information bits, its length must be that of the PCBC

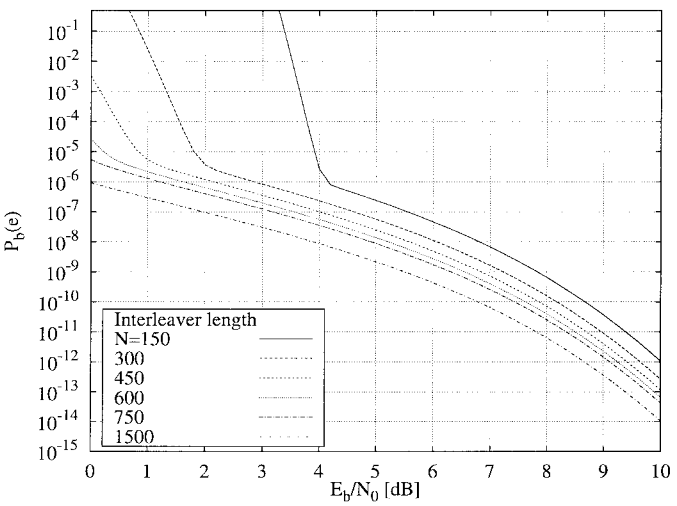

Fig. 12. Analytical bounds for the serially concatenated convolutional code SCCC4.

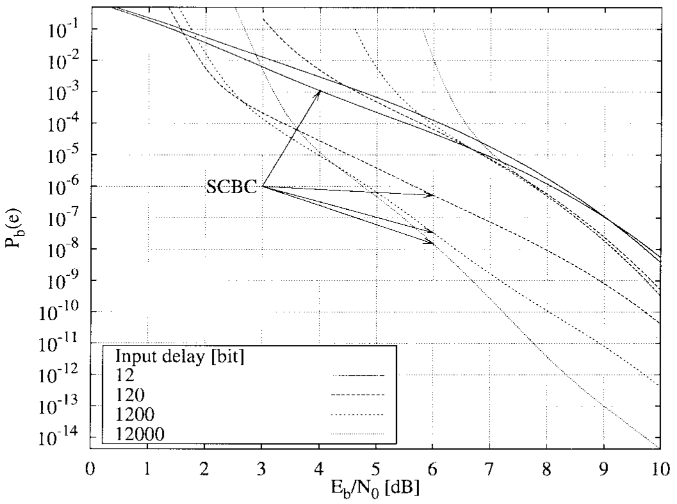

Fig. 13. Comparison of serially concatenated block codes and parallel concatenated block codes with various interleaver lengths, chosen so as to yield the same input decoding delay.

divided by R_c^o. As an example, to obtain a delay equal to 12 input bits, we must choose an interleaver length $N = 3m = 12$ for the PCBC, and $N = 7m = 12/R_c^o = 21$ for the SCBC.

The results are reported in Fig. 13, where we plot the bit error probability bounds versus the signal-to-noise ratio E_b/N_0 for various input delays. The results show that, for low values of the input delay, the performance is almost the same. On the other hand, increasing the delay (and thus the interleaver length N) yields a significant interleaver gain for the SCBC, and almost no gain for the PCBC. The difference in performance is 3 dB at $P_b(e) = 10^{-6}$ in favor of the SCBC.

B. Serially and Parallel Concatenated Convolutional Codes

To obtain a fair comparison, we have chosen the following PCCC and SCCC: the PCCC is a rate $1/3$ code obtained concatenating two equal rate $1/2$, four-state systematic recursive convolutional encoders with generator matrix as in Table III, first row. The SCCC is a rate $1/3$ code using as outer

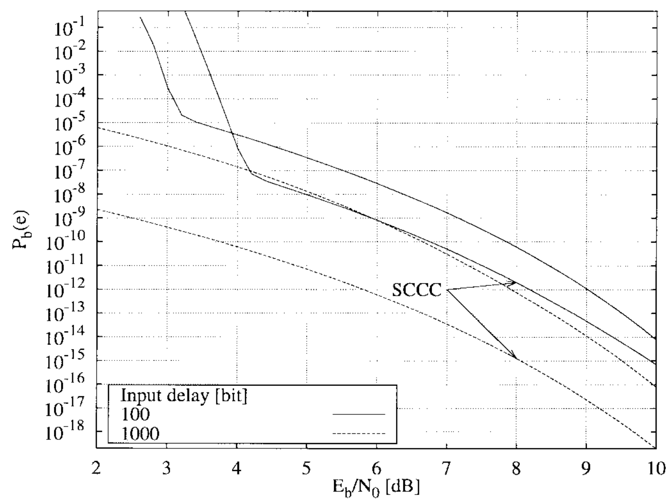

Fig. 14. Comparison of serially concatenated convolutional codes and parallel concatenated convolutional codes with four-state convolutional codes.

code the same rate $1/2$, four-state code as in the PCCC, and, as inner encoder, a rate $2/3$, four-state systematic recursive convolutional encoder with generator matrix as in Table III, third row. Also in this case, the interleaver lengths have been chosen so as to yield the same decoding delay, due to the interleaver, in terms of input bits. The results are reported in Fig. 14, where we plot the bit error probability versus the signal-to-noise ratio E_b/N_0 for various input delays.

The results show the great difference in the interleaver gain. In particular, the PCCC shows an interleaver gain going as N^{-1}, whereas the interleaver gain of the SCCC, as from (34), goes as $N^{-(d_f^o+1)/2} = N^{-3}$, being the free distance of the outer code equal to 5. This means, for $P_b(e) = 10^{-11}$, a gain of more than 2 dB in favor of the SCCC.

Previous comparisons have shown that serial concatenation is advantageous with respect to parallel concatenation, in terms of ML performance. For medium-long interleaver lengths, this significant result remains a theoretical one, as ML decoding is an almost impossible achievement. For parallel concatenated codes ("turbo codes"), iterative decoding algorithms have been proposed, which yield performance close to optimum, with limited complexity. In the following section, we will present a new iterative decoding scheme capable of decoding serially concatenated codes, and prove, with several examples of applications, that the performance gain with respect to parallel concatenation is maintained.

V. Iterative Decoding of Serially Concatenated Codes

In this section, we present a new iterative algorithm for decoding serially concatenated codes, with complexity not significantly higher than that needed to separately decode the two CC's. Because of the importance in applications, all examples will refer to SCCC's, although the decoding algorithm can be applied to SCBC's as well.

The core of the new decoding procedure consists of a block called SISO (Soft-Input Soft-Output). It is a four-port device, which accepts as inputs the probability distributions (or the

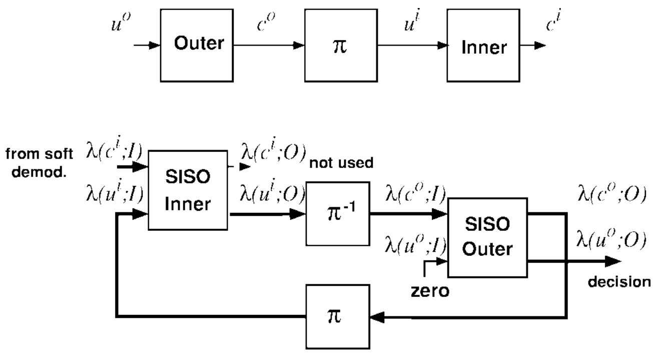

Fig. 15. Block diagrams of the encoder and iterative decoder for serially concatenated convolutional codes.

corresponding likelihood ratios) of the information and code symbols labeling the edges of the code trellis, and forms as outputs an update of these probability distributions based upon the code constraints. The block SISO is used within the iterative decoding algorithm as shown in Fig. 15, where we also show the block diagram of the encoder to clarify the notations.

We will first explain in words how the algorithm works, according to the blocks of Fig. 15. Following that, we will give the input–output relationships of the block SISO.

The symbols $\lambda(\cdot; I)$ and $\lambda(\cdot; O)$ at the input and output ports of SISO refer to the logarithmic likelihood ratios (LLR's),[10] unconstrained when the second argument is I, and modified according to the code constraints when it is O. The first argument u refers to the information symbols of the encoder, whereas c refers to code symbols. Finally, the superscript o refers to the outer encoder, and i to the inner encoder. The LLR's are defined as

$$\lambda(x; \cdot) \triangleq \log\left[\frac{P(x; \cdot)}{P(x_{\text{ref}}; \cdot)}\right]. \tag{36}$$

When x is a binary symbol, "0" or "1," x_{ref} is generally assumed to be the "1." When x belongs to an L-ary alphabet, we can choose as x_{ref} each one of the L symbols.

Differently from the iterative decoding algorithm employed for turbo decoding, in which only the LLR's of information symbols are updated, we must update here the LLR's of both information and code symbols based on the code constraints.

During the first iteration of the SCCC algorithm,[11] the block "SISO Inner" is fed with the demodulator soft outputs, consisting of the LLR's of symbols received from the channels, i.e., of the code symbols of the inner encoder. The second input $\lambda(u^i; I)$ of the SISO Inner is set to zero during the first iteration, since no *a priori* information is available on the input symbols u^i of the inner encoder.

The LLR's $\lambda(c^i; I)$ are processed by the SISO algorithm, which computes the *extrinsic* LLR's of the information symbols of the inner encoder $\lambda(u^i; O)$ conditioned on the inner code constraints. The extrinsic LLR's are passed through the

[10] When the symbols are binary, only one LLR is needed; when the symbols belong to an L-ary alphabet, $L-1$ LLR's are required.

[11] To simplify the description, we assume for now that the interleaver acts on symbols instead of bits. In the actual decoder, we deal with bit LLR's and bit interleaver, as it will be seen later.

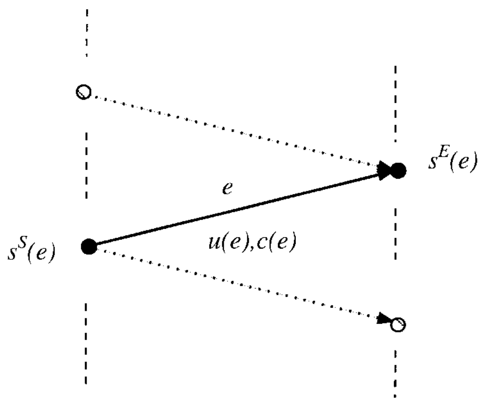

Fig. 16. Trellis section defining the notations used for the description of the SISO algorithm.

inverse interleaver (a block labeled "π^{-1}"), whose outputs correspond to the LLR's of the code symbols of the outer code, i.e.,

$$\pi^{-1}[\lambda(u^i; O)] = \lambda(c^o; I).$$

These LLR's are then sent to the block "SISO Outer" in its upper entry, which corresponds to code symbols. The SISO Outer, in turn, processes the LLR's $\lambda(c^o; I)$ of its unconstrained code symbols, and computes the LLR's of both code and information symbols based on the code constraints. The input $\lambda(u^o; I)$ of the SISO Outer is always set to zero, which implies assuming equally likely transmitted source information symbols. The output LLR's of information symbols (which yield the *a posteriori* LLR's of the SCCC information symbols) will be used in the final iteration to recover the information bits. On the other hand, the LLR's of outer code symbols, after interleaving are fed back to the lower entry (corresponding to information symbols of the inner code) of the block SISO Inner to start the second iteration. In fact, we have

$$\pi[\lambda(c^o; O)] = \lambda(u^i; I).$$

A. The Input–Output Relationships for the Block SISO

The block SISO has been described in [19]. It represents a slight generalization of the BCJR algorithm (see [20], [21], and [24]). Here, we will only recall for completeness its input–output relationships. They will refer, for notations, to the trellis section of the trellis encoder, assumed to be time-invariant as we deal with convolutional codes, shown in Fig. 16, where the symbol e denotes the trellis edges, and where we have identified the information and code symbols associated to the edge e as $u(e)$, $c(e)$, and the starting and ending states of the edge e as $s^S(e)$, $s^E(e)$, respectively.

The block SISO works at *symbol* level, i.e., for an (n, p) convolutional code, it operates on information symbols u belonging to an alphabet with size 2^p and on code symbols belonging to an alphabet with size 2^n. We will give the general input–output relationships, valid for both outer and inner SISO's, assuming that the information and code symbols are defined over a finite time index set $[1, \cdots, K]$.

At time k, $k = 1, \cdots, K$, the output extrinsic LLR's are computed as

$$\lambda_k(c; O) = \max^*_{e:\, c(e)=c} \{\alpha_{k-1}[s^S(e)] + \lambda_k[u(e); I] + \beta_k[s^E(e)]\} + h_c \quad (37)$$

$$\lambda_k(u; O) = \max^*_{e:\, u(e)=u} \{\alpha_{k-1}[s^S(e)] + \lambda_k[c(e); I] + \beta_k[s^E(e)]\} + h_u. \quad (38)$$

The name *extrinsic* given to the LLR's computed according to (37) and (38) derives from the fact that the evaluation of $\lambda_k(c; O)$ (and of $\lambda_k(u; O)$) does not depend on the corresponding simultaneous input $\lambda_k(c; I)$ (and $\lambda_k(u; I)$), so that it can be considered as an update of the input LLR based on information coming from all homologous symbols in the sequence, except the one corresponding to the same symbol interval.

The quantities $\alpha_k(\cdot)$ and $\beta_k(\cdot)$ in (37) and (38) are obtained through the *forward* and *backward* recursions, respectively, as

$$\alpha_k(s) = \max^*_{c:\, s^E(e)=s} \{\alpha_{k-1}[s^S(e)] + \lambda_k[u(e); I] + \lambda_k[c(e); I]\}, \quad k = 1, \cdots, K-1 \quad (39)$$

$$\beta_k(s) = \max^*_{e:\, s^S(e)=s} \{\beta_{k+1}[s^E(e)] + \lambda_{k+1}[u(e); I] + \lambda_{k+1}[c(e); I]\}, \quad k = K-1, \cdots, 1 \quad (40)$$

with initial values

$$\alpha_0(s) = \begin{cases} 0, & s = S_0 \\ -\infty, & \text{otherwise} \end{cases}$$

$$\beta_K(S_i) = \begin{cases} 0, & s = S_K \\ -\infty, & \text{otherwise.} \end{cases}$$

The quantities h_c, h_u in (37) and (38) are normalization constants.

The operator $\max^*$ performs the following operation:

$$\max^*_j(a_j) \triangleq \log \left[\sum_{j=1}^{J} e^{a_j} \right]. \quad (41)$$

This operation, a crucial one in affecting the computational complexity of the SISO algorithm, can be performed in practice (see [22] and [23]) as

$$\max^*_j(a_j) = \max_j (a_j) + \delta(a_1, a_2, \cdots, a_J) \quad (42)$$

where $\delta(a_1, a_2, \cdots, a_J)$ is a correction term that can be computed recursively using a single-entry lookup table [22], [23].

The previous description of the iterative decoder assumed that all operations were performed at *symbol* level. Quite often, however, the interleaver operates at *bit* level to be more effective. This is the case, for example, of all results presented in Sections I–III.

Thus to perform bit interleaving, we need to transform the symbol extrinsic LLR's obtained at the output of the first SISO into extrinsic bit LLR's, before they enter the de-interleaver. After de-interleaving, the bit LLR's need to be compacted into symbol LLR's before entering the second SISO block, and so on. These operations are performed under the assumption that the bits forming a symbol are independent.

Assuming an (n, p) code, and denoting with $\boldsymbol{u} = [u_1, \cdots, u_p]$ the information symbol formed by p information bits, the extrinsic LLR λ_i of the ith bit u_i within the symbol $\boldsymbol{u}$ is obtained as

$$\lambda_i(u; O) = \overset{*}{\max_{\boldsymbol{u}:\, u_i=0}}[\lambda(\boldsymbol{u}; O) + \lambda(\boldsymbol{u}; I)] - \overset{*}{\max_{\boldsymbol{u}:\, u_i=1}}[\lambda(\boldsymbol{u}; O) + \lambda(\boldsymbol{u}; I)] - \lambda_i(u; I). \quad (43)$$

Conversely, the extrinsic LLR of the symbol $\boldsymbol{u}$ is obtained from the extrinsic LLR's of its component bits u_i as

$$\lambda(\boldsymbol{u}) = \sum_{i=1}^{p} \lambda_i(u_i). \quad (44)$$

The previous description should have made clear that the SISO algorithm requires the whole sequence to be received before starting. The reason is due to the backward recursion that starts from the (supposed known) final trellis state. As a consequence, its practical application is limited to the case where the duration of the transmission is short (K small), or, for K long, when the received sequence can be segmented into independent consecutive blocks, like for block codes or convolutional codes with trellis termination [14]. It cannot be used for continuous decoding. This constraint leads to a frame rigidity imposed on the system, and also reduces the overall code rate, because of trellis termination.

A more flexible decoding strategy is offered by modifying the algorithm in such a way that the SISO module operates on a fixed memory span, and outputs the smoothed probability distributions after a given delay D. This algorithm, which we have called the *sliding window soft-input soft-output* (SW-SISO) algorithm, is fully described in [23]. In the following simulation results, the SW-SISO algorithm has been applied.

B. Applications of the Decoding Algorithm

We will now use the decoding algorithm to confirm the design rules presented before, and to show the behavior of SCCC in the region of low signal-to-noise ratios (below cutoff rate). Since in this region analytical bounds fail to give significant results, no meaningful quantitative comparisons can be performed between simulated and analytical performance. However, we will show that the hierarchy of the simulation results agrees with the design considerations that had been based on the analysis.

The following aspects will be considered:

- behavior of the decoding algorithm versus the number of decoding iterations;
- behavior of the decoding algorithm versus the interleaver length;
- the effect of choosing a nonrecursive inner code;
- SCCC behavior for very low signal-to-noise ratios, to see how close serial concatenation can get to theoretical Shannon bound;

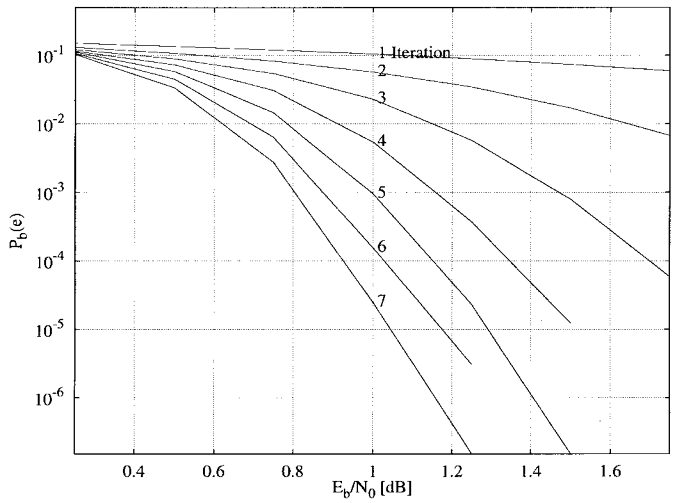

Fig. 17. Simulated bit error probability versus the number of iterations for a rate $1/3$ serially concatenated convolutional code obtained concatenating a 4-state rate $1/2$ recursive convolutional code and a 4-state rate $2/3$ recursive convolutional code. The concatenated code is the code SCCC1 of Table III. The decoding delay in terms of input bits due to the interleaver is 1024.

- comparison between SCCC's and PCCC's (turbo codes) for the same value of decoding delay imposed by the two schemes on the input bits.

For all simulated SCCC's, we have used purely random interleavers.

1) Simulated Coding Gain Versus Number of Iterations: Consider the rate $1/3$ SCCC1 of Table II. It employs two 4-state recursive convolutional encoders, the first (outer code) with rate $1/2$ and the second (inner code) with rate $2/3$, joined by an interleaver of length $N = 2048$. Since the interleaver operates on coded sequences produced by the outer rate $1/2$ encoder, its length of 2048 bits corresponds to a delay of 1024 information bits. The simulation results are shown in Fig. 17 in terms of bit error probability versus E_b/N_0 for a number of iterations N_I ranging from 1 to 7. The nice convergence of the decoding algorithm is manifest.

2) The Effect of a Nonrecursive Inner Encoder: The analysis of Section III came to the conclusion that a nonrecursive inner encoder should yield little interleaver gains. To confirm this theoretical prediction by simulation results, we plot in Fig. 18 the bit error probability versus the input decoding delay obtained by simulating the concatenated code SCCC2 of Table III. This code uses as inner encoder a 4-state nonrecursive encoder. The curves refer to a signal-to-noise ratio $E_b/N_0 = 1.5$ dB, and to a number of iterations N_I ranging from 1 to 10. It is evident that the bit error probability reaches the floor of 10^{-5} for a decoding delay greater than or equal to 1024, so that no interleaver gain takes place beyond this point. For comparison, we report in Fig. 19 the results obtained for the code SCCC3 of Table III. The curves refer to a signal-to-noise ratio of 0.75 dB, and show the interleaver gain predicted by the analysis.

3) Approaching the Theoretical Shannon Limit: We show here the capabilities of SCCC's of yielding results close to the Shannon capacity limit. To this purpose, we have chosen a rate $1/4$ concatenated scheme with very long interleaver,

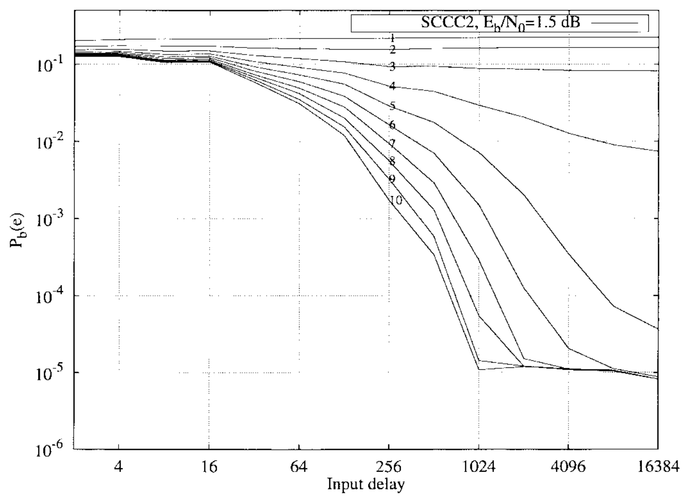

Fig. 18. Simulated performance of concatenated code SCCC2 of Table III. The bit error probability is plotted versus input decoding delay for different number of iterations. The signal-to-noise ratio is 1.5 dB.

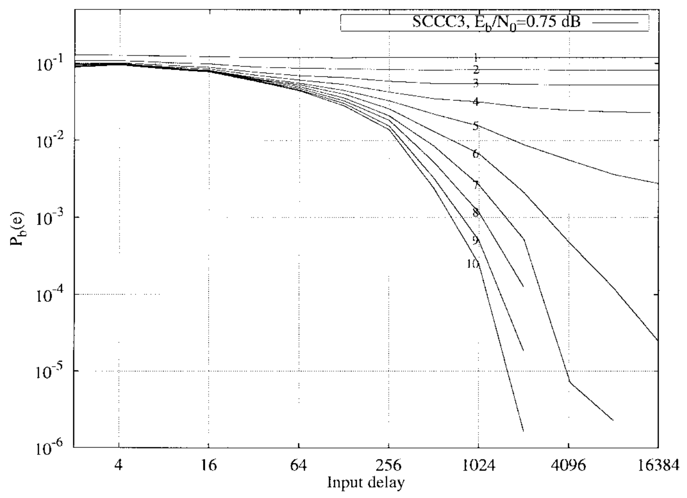

Fig. 19. Simulated performance of concatenated code SCCC3 of Table III. The bit error probability is plotted versus input decoding delay for different number of iterations. The signal-to-noise ratio is 0.75 dB.

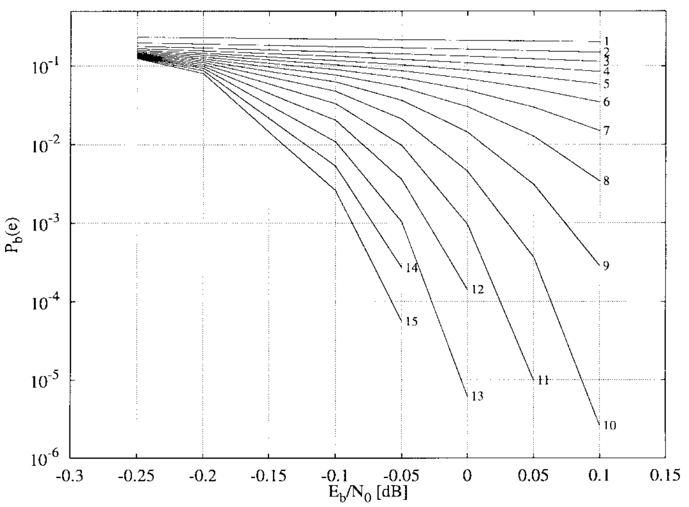

Fig. 20. Simulated performance of a rate $1/4$ serially concatenated code obtained with two eight-state constituent codes and an interleaver yielding an input decoding delay equal to 16 384.

corresponding to an input decoding delay of 16 384. The constituent codes are 8-state codes: the outer encoder is nonrecursive, and the inner encoder is a recursive encoder. Their generating matrices are

$$G_o(D) = [1+D, 1+D+D^3]$$

$$G_i(D) = [1, \frac{1+D+D^3}{1+D}]$$

respectively. Note the feedback polynomial $(1+D)$ in the generator matrix of the inner encoder, which eliminates error events with odd input weights. The results in terms of bit error probability versus signal-to-noise ratio for different number of iterations are presented in Fig. 20. They show that the decoding algorithm works at $E_b/N_0 = -0.05$ dB, at 0.76 dB from the Shannon capacity limit (which is in this case equal to -0.817 dB), with very limited complexity (remember that we are using two rate $1/2$ codes with eight states).

4) Comparison Between Serially and Parallel Concatenated Codes: Previous analytical results showed that serial concatenation can yield significantly higher interleaver gains and steeper asymptotic slope of the error probability curves. To check if these advantages are retained when the codes are iteratively decoded at very low signal-to-noise ratios, we have simulated the behavior of SCCC's and PCCC's in equal system conditions: the concatenated code rate is $1/3$, the CC's are 4-state recursive encoders (rates $1/2 + 1/2$ for PCCC's, and rates $1/2 + 2/3$ for the SCCC's), and the decoding delays in terms of input bits are 256, 1024, and 16 384, respectively. In Fig. 21, we report the results, in terms of bit error probability versus signal-to-noise ratio, for the case of a decoding delay equal to 256, after three and nine decoding iterations. As it can be seen from the curves, the PCCC outperforms the SCCC for high values of the bit error probabilities. Below 10^{-3} (for nine iterations), the SCCC behaves significantly better, and does not present the floor.[12] The improvement of SCCC with respect to PCCC becomes more visible in Fig. 22, which refers to a delay of 1024 and to three and seven iterations of the decoding algorithm. For bit error probabilities lower than 10^{-2}, the SCCC outperforms PCCC. In particular, the absence of error floor is striking here. At 10^{-4}, SCCC has an advantage of 0.7 dB with seven iterations. Finally, in Fig. 23, we report the results for an input decoding delay of 16 384 and six and nine decoding iterations. In this case, the crossover between PCCC and SCCC happens around 10^{-5}. The advantage of SCCC at 10^{-6} is 0.5 dB with nine iterations.

As a conclusion, we can say that the advantages obtained for signal-to-noise ratios above the cutoff rate, where the union bounds can be safely applied, are retained also in the region between channel capacity and cutoff rate. Only when the system interest focuses on high values of bit error probability (the threshold depending on the interleaver size) the PCCC are to be preferred. PCCC's, however, present a floor to the bit error probability, which, in the most favorable case seen

[12] It is customary in the turbo codes literature to call "error floor" what is actually a sensible change of slope of the performance curve.

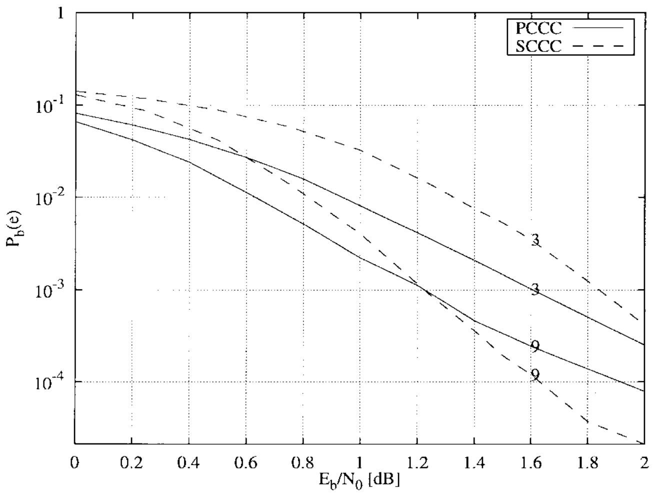

Fig. 21. Comparison of two rate $1/3$ parallel concatenated convolutional code and serially concatenated convolutional code. The parallel concatenated convolutional code is obtained concatenating two equal rate $1/2$ 4-state codes (first code in Table III); the serially concatenated convolutional code is the code SCCC1 of Table II. The curves refer to three and nine iterations of the decoding algorithm and to an equal input decoding delay of 256.

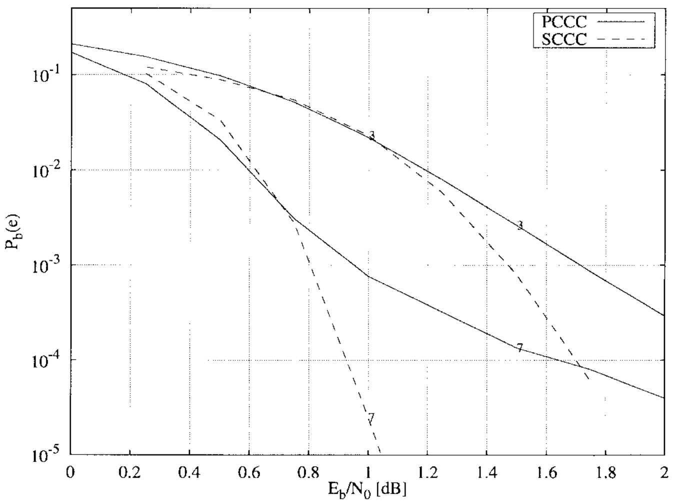

Fig. 22. Comparison of two rate $1/3$ parallel concatenated convolutional code and serially concatenated convolutional code. The parallel concatenated convolutional code is obtained concatenating two equal rate $1/2$ 4-state codes (first code in Table III); the serially concatenated convolutional code is the code SCCC1 of Table II. The curves refer to three and seven iterations of the decoding algorithm and to an equal input decoding delay of 1024.

above, lies around 10^{-6}. This floor is absent, or, at least, much lower, in the case of SCCC.

Finally, it must be recognized that the constituent codes design rules presented in Section III are based on union bound considerations, and thus yield optimum SCCC's above the cutoff rate. For system applications aiming at very low signal-to-noise ratios, close to the channel capacity (as, for example, in deep-space communications), a general statement is that complex CC's should be avoided, and CC's with low number of states (4–16) should be used. So far, finding the best codes to operate in the region of very low signal-to-noise ratios, has been a matter of feeling dictated by experience and, literally, case-by-case "crafting."

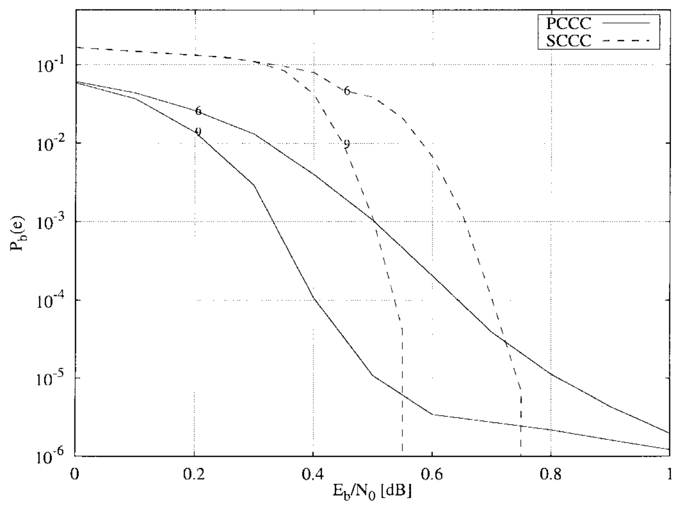

Fig. 23. Comparison of two rate $1/3$ parallel concatenated convolutional code and serially concatenated convolutional code. The parallel concatenated convolutional code is obtained concatenating two equal rate $1/2$ 4-state codes (first code in Table III); the serially concatenated convolutional code is the code SCCC1 of Table II. The curves refer to six and nine iterations of the decoding algorithm and to an equal input decoding delay of 16 384.

VI. Conclusions

Serially concatenated codes with interleaver, a concept building on classical concatenated codes and parallel concatenated codes known as "turbo codes," have been studied. They consists of the cascade of an *outer* encoder, an interleaver permuting the outer codewords bits, and an *inner* encoder whose input words are the permuted outer codewords. Upper bounds to the *average* maximum-likelihood bit error probability of serially concatenated block and convolutional coding schemes have been derived. Based on those bounds, we have derived design guidelines for the outer and inner codes that maximize the *interleaver gain* and the asymptotic slope of the error-probability curves. It has been shown that the interleaver gain, defined as the factor that decreases the bit error probability as a function of the interleaver size, can be made significantly higher than for turbo codes. Finally, a new, low-complexity iterative decoding algorithm that yields performance close to the Shannon limit has been illustrated. Extensive examples have been presented, and comparisons with parallel concatenated convolutional codes have been performed. The simulation results show that, in general, SCCC's have significantly lower changes of slope in the bit error probability curves than PCCC's. On the other hand, if the bit error probabilities achievable by PCCC's above the change of slope are sufficiently low, they can be reached by PCCC's at slightly lower signal-to-noise ratios than by SCCC's. As a conclusion, PCCC's seem thus better suited to approaching capacity limits for bit error probabilities above the error floor (10^{-6}–10^{-7}), whereas SCCC's are better suited to provide near error-free performance.

References

[1] G. D. Forney, Jr., *Concatenated Codes.* Cambridge, MA: MIT Press, 1966.

[2] R. H. Deng and D. J. Costello, "High rate concatenated coding systems using bandwidth efficient trellis inner codes," *IEEE Trans. Commun.*, vol. 37, pp. 420–427, May 1989.

[3] J. Hagenauer and P. Hoeher, "Concatenated Viterbi decoding," in *Proc. 4th Joint Swedish-Soviet Int. Workshop on Information Theory* (Gotland, Sweden, Studenlitteratur, Lund, Aug. 1989), pp. 29–33.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. ICC'93* (Geneva, Switzerland, May 1993), pp. 1064–1070.

[5] S. Benedetto and G. Montorsi, "Serial concatenation of block and convolutional codes," *Electron. Lett.*, vol. 32, no. 10, pp. 887–888, May 1996.

[6] ——, "Iterative decoding of serially concatenated convolutional codes," *Electron. Lett.*, vol. 32, no. 13, pp. 1186–1187, June 1996.

[7] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[8] S. Benedetto and G. Montorsi, "Unveiling turbo-codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–429, Mar. 1996.

[9] G. Poltyrev, "Bounds on the decoding error probability of binary linear codes via their spectra," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1284–1292, July 1994.

[10] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New York: McGraw-Hill, 1979.

[11] T. M. Duman and M. Salehi, "New performance bounds for turbo codes," in *Proc. GLOBECOM'97* (Phoenix, AZ, Nov. 1997).

[12] A. J. Viterbi, A. M. Viterbi, and J. Nicolas, "Perspectives on interleaved concatenated codes with iterative soft-output decoding," private communications, May 1997.

[13] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.

[14] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. ICC'95* (Seattle, WA, June 1995).

[15] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electron. Lett.*, vol. 31, no. 3, pp. 156–158, Feb. 1995.

[16] G. D. Forney, Jr., "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720–738, Sept. 1970.

[17] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electron. Lett.*, vol. 32, no. 5, Feb. 1996.

[18] P. Thitimajshima, "Systematic recursive convolutional codes and their application to parallel concatenation," Ph.D. dissertation, Univ. de Bretagne Occidentale, Dec. 1993, in French.

[19] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Commun. Lett.*, vol. 1, no. 1, pp. 22–24, Jan. 1997.

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

[21] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072–1091, July 1996.

[22] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. ICC'95* (Seattle, WA, June 1995), pp. 1009–1013.

[23] 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," *Europ. Trans. Telecommun.*, April 1998, invited paper.

[24] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.