A Novel Technique for the Evaluation of the Transfer Function of Parallel Concatenated Convolutional Codes

Ioannis Chatzigeorgiou [1], Miguel R.D. Rodrigues [1], Ian J. Wassell [1], Rolando Carrasco [2]

[1] Computer Laboratory, University of Cambridge, UK. E-mail: {ic231, mrdr3, ijw24}@cam.ac.uk
[2] Dept. of EE&C Engineering,University of Newcastle upon Tyne, UK. E-mail: r.carrasco@ncl.ac.uk

Abstract

Turbo codes, in the form of parallel concatenated convolutional codes, consist of two recursive systematic convolutional encoders separated by an interleaver. Due to the presence of the interleaver, each constituent convolutional encoder accepts as input a block of information bits with a size equal to that of the interleaver rather than a continuous stream of information bits. By determining the transfer function of each terminated constituent convolutional code, which can be seen as a convolutional block code, an upper bound to the bit error rate performance of the turbo code is readily calculated. In this paper, we briefly present the conventional techniques for evaluating the transfer function of the convolutional block code and we propose a novel technique, according to which the state diagram of the convolutional code is modified so as to allow the direct evaluation of the transfer function of the convolutional block code.

1 Introduction and Motivation

Turbo codes, originally conceived by Berrou *et al*. over a decade ago [1, 2], are widely known for their astonishing performance in the additive white Gaussian noise (AWGN) channel. Methods to evaluate an upper bound to the bit error probability (BEP) of a parallel-concatenated coding scheme were proposed by Divsalar *et al*. [3] as well as Benedetto and Montorsi [4]. In addition, guidelines for the optimal design of the constituent convolutional codes were presented in [5]. An upper bound to the BEP has also been derived for the case of serially concatenated coding schemes [6]. In all previous cases the analytical bounding technique requires knowledge of the constituent codes. A more abstract approach, according to which the turbo code is seen as a whole and only input words of weight 2 and 3 are taken into account in the expression for the BEP, is presented in [7]. All methods provide accurate results for high values of E_b/N_0, however the approaches in [3] and [4] are more suitable for the investigation and selection of constituent codes with the aim of improving the overall performance of the turbo code.

The analytical techniques presented in [3] and [4] introduce the concept of the input-output weight enumerating function (IOWEF) associated with the terminated constituent convolutional codes of the turbo encoder. In [3] the evaluation of IOWEF involves a matrix inversion following the computation of the state transition matrix of the convolutional code. In [4] the IOWEF is calculated after evaluating a modified version of the transfer function of the convolutional code.

The motivation for this paper is to propose a different method for the evaluation of the transfer function of the terminated convolutional code. More specifically, the transfer function of the terminated convolutional code is computed using an augmented version of the modified state diagram of the original convolutional code. Furthermore, we demonstrate how to derive only those terms of the transfer function associated with a specific block length as well as a specific range of input and output weights.

In Section 2 of this paper, we briefly describe the conventional evaluation of the transfer function of a convolutional code, while in Section 3 we show how to determine the transfer function of the terminated convolutional code. In Section 4, we present our approach for the evaluation of the transfer function of the terminated convolutional code. In Section 5 we provide expressions for the performance of a turbo code. We conclude by presenting simulation results in Section 6 and a summary of the contributions in Section 7.

2 Evaluation of the Transfer Function of Convolutional Codes

A detailed description of the properties of convolutional codes as well as expressions required to evaluate the performance of the maximum likelihood (ML) decoder, are presented in [8]. In particular it is shown that an upper bound to the BEP of the ML decoder for binary phase shift keyed (BPSK) modulation in an AWGN channel can be obtained if the transfer function of the convolutional code is known.

A generic form of the transfer function $T^C(W,D,L)$ of a convolutional code is:

$$T^C(W,D,L) = \sum_{w,d,l} T_{w,d,l} W^w D^d L^l \,, \qquad (1)$$

where $T_{w,d,l}$ denotes the number of paths that start from the zero state and remerge with the zero state after l steps, i.e., their length is l, and are associated with an input sequence of weight w, and an output sequence of weight d.

An example of a recursive convolutional encoder with code rate 1/2, recursive generator polynomial 5 and forward generator polynomial 7, i.e., RSC(1,7/5), is shown in Fig.1(a). The state diagram of the code is illustrated in Fig.1(b). A branch corresponds to the transition from a state to another state. Each branch is labelled by the input word that caused the transition and the output word generated by the transition.

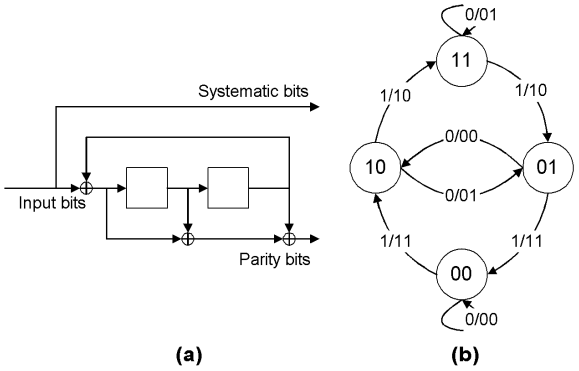

Fig. 1 Block diagram (a) and state diagram (b) of RSC(1,7/5)

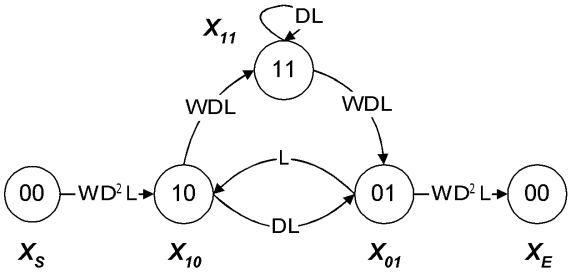

Fig. 2 Modified state diagram of RSC(1,7/5)

The modified state diagram of the convolutional code, presented in Fig.2, can be used to obtain its transfer function. In order to avoid circulation around the self-loop, we split the state 00 into two separate states, the start state X_S and the end state X_E. Furthermore, we label the branches according to the Hamming weight of the input and output words. The exponent of W corresponds to the weight of the input word whereas the exponent of D corresponds to the weight of the output word. Each branch also has term L, which represents a time step. We express each state of the diagram as a function of the other states, so as to obtain the state equations. Upon solving these equations for the ratio X_E/X_S, we obtain the transfer function for RSC(1,7/5):

$$\begin{aligned} T^C(W,D,L) &= W^2D^5L^3 + W^4D^6L^4 \\ &\quad + (W^2D^6 + W^4D^7)L^5 + \ldots \end{aligned} \qquad (2)$$

which tells us that there is one path of length 3 generated by an input sequence of weight 2 which produces an output sequence of weight 5 (i.e., $W^2D^5L^3$), and so on.

3 Derivation of the Transfer Function of Convolutional Block Codes

A parallel concatenated convolutional code (PCCC) is formed by an interleaver of length N and two convolutional encoders [1, 2]. For simplicity, we assume identical constituent encoders that generate the same convolutional code. Since each constituent code is terminated after N input bits, we refer to it as convolutional block code C. It was shown in [3] and [4] that the performance of a PCCC depends upon the transfer function, or equivalently on the input-output weight enumerating function (IOWEF) of the convolutional block code C, which assumes the form:

$$\begin{aligned} B^C(W,D) &= \sum_{w=1}^{N_w} \sum_{d=1}^{N_D} B_{w,d} W^w D^d \\ &= \sum_{w,d} B_{w,d} W^w D^d \end{aligned} \qquad (3)$$

where $B_{w,d}$ denotes the number of codewords with weight d generated by an input sequence of weight w.

The trellis of the convolutional block code, whose input is a block of N bits, is the truncation at step N of the infinite length trellis of the original convolutional code. More specifically, the transfer function $T^C(W,D,L)$ of a convolutional code provides all paths that start from the zero state, diverge from the all-zero path at step 1 and at some point remerge with the zero state and remain at it. The IOWEF $B^C(W,D)$ of the convolutional block code provides all paths of length up to N that start from the zero state, can remerge with and diverge from the zero state more than once and terminate at the zero state.

In the following two subsections, the techniques for the evaluation of the IOWEF of the convolutional block code, as described in [3] and [4], are briefly presented.

3.1 Divsalar's Technique

Divsalar *et al.* [3] transform the information contained in the state diagram of the convolutional code into the transition matrix $\mathbf{A}(W,D,L)$. The information in the state diagram of the 4-state RSC(1,7/5) in Fig.1(b), can be summarised by the 4×4 transition matrix, $\mathbf{A}(W,D,L) = [\alpha_{ij,1}(W,D,L)]$, as follows:

$$\mathbf{A}(W,D,L)=\begin{bmatrix} L & 0 & WD^2L & 0 \\ WD^2L & 0 & L & 0 \\ 0 & DL & 0 & WDL \\ 0 & WDL & 0 & DL \end{bmatrix} \quad (4)$$

where $\alpha_{ij,1}(W,D,L)$ contains all the information (i.e., input weight W, output weight D and length L) conveyed by the transition from state i to state j after one time step. The third index, which in this case is 1, denotes the number of time steps and is also equal to the exponent of L of the non-zero terms. Moreover, each input bit to the encoder of our example causes the trellis of the code to expand by one additional time step, therefore the third index is also equal to the block size of the input sequence.

For a block of size N, the IOWEF of the convolutional block code can be found by considering only the (0,0) element of the matrix $\mathbf{A}^N(W,D,L)=[\alpha_{ij,N}(W,D,L)]$, i.e.,

$$B^C(W,D)=\alpha_{00,N}(W,D,L)\Big|_{L=1}. \quad (5)$$

The (0,0) element $\alpha_{00,N}(W,D,L)$, is a summation of terms that correspond to all paths that start from and terminate at the zero state after N time steps. Consequently all terms of $\alpha_{00,N}(W,D,L)$ are functions of L^N. By setting L=1, we derive the IOWEF $B^C(W,D)$ of the convolutional block code.

Instead of calculating the N-th power of $\mathbf{A}(W,D,L)$, we compute the sum of all $\mathbf{A}^k(W,D,L)$ associated with all possible block sizes, i.e., k = 0,1,…∞ :

$$\mathbf{I}+\mathbf{A}(W,D,L)+\mathbf{A}^2(W,D,L)+\ldots=(\mathbf{I}-\mathbf{A}(W,D,L))^{-1} \quad (6)$$

Consequently, the (0,0) element of $(\mathbf{I}\text{-}\mathbf{A}(W,D,L))^{-1}$ = $[\alpha_{ij}(W,D,L)]$ will be the sum of all (0,0) elements associated with all block sizes:

$$\alpha_{00}(W,D,L)=\sum_{k=0}^{\infty}\alpha_{00,k}(W,D,L). \quad (7)$$

For a block of size k=N, we use (7) to derive element $\alpha_{00,N}(W,D,L)$ by means of the recursion described in [3]. The IOWEF of the convolutional block code is found by substituting $\alpha_{00,N}(W,D,L)$ in (5).

3.2 Benedetto's and Montorsi's Technique

The technique that Benedetto and Montorsi [4] proposed, requires the modification of the transfer function $T^C(W,D,L)$ of the convolutional code, as follows:

$$T^C(W,D,L,\Omega)=\sum_{w,d,l,n}T_{w,d,l,n}W^wD^dL^l\Omega^n, \quad (8)$$

where $T_{w,d,l,n}$ is the number of paths (i.e., codewords) that start from the zero state and remerge with the zero state n times, having length $l\le N$, and are associated with an input sequence of weight w and an output sequence of weight d. When a path merges with and then diverges from the zero state, it cannot stay at the zero state for more than one time-step. The transfer function $T^C(W,D,L,\Omega)$ can be evaluated by elaborating on the algorithm described in [9].

Since convolutional codes are linear, the set of input and output weights of all paths from the path associated with the all-zero input sequence is the same as the set of weights of all paths with respect to any other path. Therefore, without loss of generality, we assume that the all-zero sequence is the input to the encoder. The all-zero input sequence forces the encoder to stay at the zero state and generate an all-zero output sequence. The corresponding path in the trellis of the code is known as the all-zero path.

In this case, $T^C(W,D,L,\Omega)$ provides those codewords associated with n remergings with the all-zero path, which is equivalent to n successive error events. However, it does not account for the codewords with zeros before, after or between the error events. It was shown in [4] that the total number of codewords with zeros before, after or between the n error events of total length $l\le N$, can be obtained from a single codeword associated with n successive errors of length l. The total number of codewords is given by the following expression:

$$K[l,n]=\binom{N-l+n}{n}=\frac{(N-l+n)!}{(N-l)!n!} \quad (9)$$

Therefore, for a given input weight w, codeword weight d, length l and n remergings, there are $T_{w,d,l,n}$ codewords associated with n successive error events and $K[l,n]\cdot T_{w,d,l,n}$ codewords with zeros before, after or between the n error events. Consequently, the total number of codewords $B_{w,d}$ with weight d generated by an input word of weight w can be obtained from:

$$B_{w,d}=\sum_{l,n}K[l,n]\cdot T_{w,d,l,n} \quad (10)$$

The IOWEF $B^C(W,D)$ of the convolutional block code is obtained by substituting $B_{w,d}$ from (10) into (3).

4 Proposed Technique

In this section we suggest a modification of the state diagram of the original convolutional code from which the IOWEF of a convolutional block code can be directly derived, instead of evaluating the modified transfer function of the convolutional code and then performing additional computations, as described in [4] and in the previous section.

4.1 State Diagram of a Convolutional Block Code

For consistency, we continue to pursue the example of Fig.1. By adding a new state X_{00} as well as the circulation loop, the state diagram of a convolutional block code shown in Fig.3 can be obtained from the modified state diagram presented

previously in Fig.2. These additions enable an indefinite number of remergings with the zero state for an indefinite period of time. The new "augmented" diagram in Fig.3 also has a start state X_S from which branches only emerge, and an end state X_E where branches only terminate.

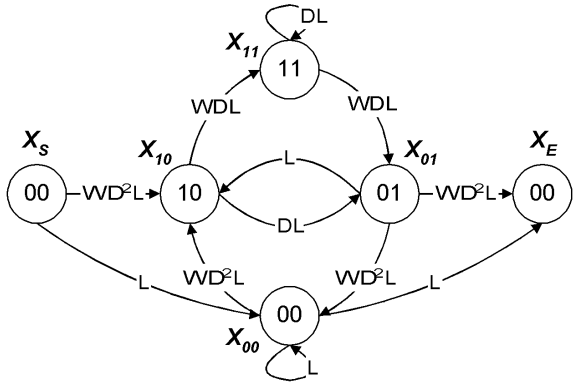

Fig. 3 Augmented state diagram for the computation of the IOWEF of the convolutional block code RSC(1, 7/5)

As usual, we derive the state equations and we solve them for the ratio X_E/X_S. The result will be a sum of two terms:

$$\frac{X_E}{X_S} = f(L) + f(W,D,L) \tag{11}$$

The first term $f(L)$ is a function of L only and represents the set of all paths with zero input weight and zero codeword weight, i.e., all the all-zero paths for different block lengths. These paths start from state X_S, stay at state X_{00} for an indefinite number of steps by circulating around the self-loop and finally terminate at state X_E. The all-zero paths are not of interest in the computation of $B^C(W,D)$ and are ignored.

The second term $f(W,D,L)$ represents the set of all paths that start from the zero state and end at the zero state for various block lengths. It can be expressed as:

$$f(W,D,L) = \frac{Y(W,D,L)}{1 - X(W,D,L)}, \tag{12}$$

where $Y(W,D,L)$ and $X(W,D,L)$ are polynomials of W, D and L and can be expressed as:

$$\begin{aligned} Y(W,D,L) &= \sum_{w_y,d_y,l_y} B_{w_y,d_y,l_y} W^{w_y} D^{d_y} L^{l_y} \\ X(W,D,L) &= \sum_{w_x,d_x,l_x} B_{w_x,d_x,l_x} W^{w_x} D^{d_x} L^{l_x} \end{aligned} \tag{13}$$

where B_{w_y,d_y,l_y}, B_{w_x,d_x,l_x} are integers.

Exploiting the property of binomial series:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \dots, \tag{14}$$

we obtain:

$$\begin{aligned} f(W,D,L) = {} & Y(W,D,L) \\ & + X(W,D,L) \cdot Y(W,D,L) \\ & + X^2(W,D,L) \cdot Y(W,D,L) \\ & + \dots \end{aligned} \tag{15}$$

The polynomial $f(W,D,L)$ finally assumes the form:

$$f(W,D,L) = \sum_{w,d,l} B_{w,d,l} W^w D^d L^l \tag{16}$$

where $B_{w,d,l}$ is a nonnegative integer.

If we are interested in a specific block size N, we need to consider only the terms L^l of $f(W,D,L)$ with $l=N$. We can rewrite (16) as follows:

$$\begin{aligned} f(W,D,L) &= \sum_{w,d,l} B_{w,d,l} W^w D^d L^l \\ &= \sum_{\substack{w,d,l \\ l \neq N}} B_{w,d,l} W^w D^d L^l + \sum_{\substack{w,d,l \\ l=N}} B_{w,d,l} W^w D^d L^l \\ &= f_{l \neq N}(W,D,L) + f_{l=N}(W,D,L) \end{aligned} \tag{17}$$

For a given block size N, the IOWEF $B^C(W,D)$ of the convolutional block code is equal to:

$$B^C(W,D) = f_{l=N}(W,D,L)\big|_{L=1} \tag{18}$$

4.2 Computation of the Required Terms

After the construction of the state diagram of a convolutional block code and the evaluation of the polynomials $Y(W,D,L)$ and $X(W,D,L)$, we need only to compute those terms of $f(W,D,L)$ that contain L^N instead of computing all terms that contain L^l, for an arbitrary high l, and then isolating the terms with exponent $l=N$.

In order to simplify (13), we define

$$\begin{aligned} y(W,D,l_y) &= \sum_{w_y,d_y} B_{w_y,d_y,l_y} W^{w_y} D^{d_y} \\ x(W,D,l_x) &= \sum_{w_x,d_x} B_{w_x,d_x,l_x} W^{w_x} D^{d_x} \end{aligned} \tag{19}$$

and we substitute in (13). For the sake of clarity, we drop W and D in this subsection, since they are not of importance for the discussion. Consequently, expression (13) becomes:

$$\begin{aligned} Y(L) &= \sum_{l_y} y(l_y) \cdot L^{l_y} \\ X(L) &= \sum_{l_x} x(l_x) \cdot L^{l_x} \end{aligned} \tag{20}$$

We assume that l_y and l_x take values from the sets:

$$\begin{aligned} l_y &\in \{l_{y_1}, l_{y_2}, \dots, l_{y_i}, \dots, l_{y_M}\} \\ l_x &\in \{l_{x_1}, l_{x_2}, \dots, l_{x_j}, \dots, l_{x_P}\} \end{aligned} \tag{21}$$

In order to find all terms of $X(L)$ raised to an arbitrary power k, as required by (15), we use the multinomial formula:

$$X^k(L) = \sum_{\substack{k_1,\dots,k_P \\ k_1+\dots+k_P=k}} \left[\frac{k!}{k_1! \dots k_P!} (x(l_{x_1}) L^{l_{x_1}})^{k_1} \cdot \dots \cdot (x(l_{x_P}) L^{l_{x_P}})^{k_P} \right] \tag{22}$$

or:

$$X^k(L) = \sum_{\substack{k_1,\ldots,k_P \\ k_1+\ldots+k_P=k}} \left[\frac{k!}{k_1!..k_P!} \left(x^{k_1}(l_{x_1}) \cdot \ldots \cdot x^{k_P}(l_{x_P}) \right) \cdot L^{(l_{x_1}k_1+\ldots+l_{x_P}k_P)} \right] \quad (23)$$

where the sum is taken over all nonnegative integers $k_1, k_2, \ldots, k_P$ for which $k_1 + k_2 + \ldots + k_P = k$.

The wanted terms that contain L^N are computed by following these steps:

1. We first select a value for l_y from the set in (21). Let's assume that this value is $l_{y_i} \le N$.
2. Based on (15), we are focused on $X(L)$ raised to such powers that result in terms, which contain L raised to the power of $(N - l_{y_i})$.
3. Considering (23), the wanted exponents of $X(L)$ can be found by solving for $k_1, k_2, \ldots, k_P$ the Diophantine [10] equation:

   $$l_{x_1}k_1 + l_{x_2}k_2 + \ldots + l_{x_P}k_P = (N - l_{y_i}) \quad (24)$$

   We then add the elements of each set $\{k_1, k_2, \ldots, k_P\}$ of nonnegative solutions to obtain the wanted exponents of $X(L)$, i.e.:

   $$k = k_1 + k_2 + \ldots + k_P \text{ for each set } \{k_1, \ldots, k_P\} \quad (25)$$

4. We substitute each set $\{k_1, k_2, \ldots, k_P\}$ as well as the corresponding sum k to (23) to compute only the terms that contain L raised to the power of $l_{x_1}k_1 + l_{x_2}k_2 + \ldots + l_{x_P}k_P$.
5. This procedure is repeated until all M values have been assigned to l_y.

The selected terms compose the IOWEF of the convolutional block code, $B^C(W,D)$, with $N_W = N$ and $N_D = 2N$, for a code rate of 1/2 and a block of size N.

5 Performance of Turbo Codes on the AWGN Channel

In this section, we briefly elaborate on the expressions described in [4], which describe the BEP performance of a PCCC. The IOWEF, $B^C(W,D)$, of a convolutional block code is computed as described in the previous section. The sum of those terms of the IOWEF associated with a specific input weight w, is known as the conditional weight enumerating function (CWEF), $B^C(w,D)$:

$$B^C(w,D) = \sum_d B_{w,d} D^d \quad (26)$$

The CWEF of the parallel concatenation of two encoders that implement the convolutional code C, separated by a uniform random interleaver of size N, is given by:

$$B^P(w,D) = \frac{D^{-w} \cdot (B^C(w,D))^2}{\binom{N}{w}} \quad (27)$$

The IOWEF of the PCCC, $B^P(W,D)$, is obtained from the CWEF, $B^P(w,D)$, using the following expression:

$$B^P(W,D) = \sum_w B^P(w,D) W^w = \sum_{w,d} B^P_{w,d} W^w D^d \quad (28)$$

The ML upper bound of the PCCC is then approximated by:

$$P_b \le \sum_{w,d} \frac{w}{N} B^P_{w,d} \, Q\left(\sqrt{\frac{2R_P E_b}{N_0} \cdot d} \right), \quad (29)$$

where R_P is the code rate of the PCCC.

As E_b/N_0 increases, the $Q(.)$ function approaches zero for large output weights d. Therefore at high values of E_b/N_0 the upper bound mainly depends on those terms of the IOWEF associated with low output weights. Due to the systematic nature of the turbo code, codewords of low weight d tend to be generated by input words of low weight w. If we are only interested in these 'most significant' terms, we can introduce additional constraints to the afore-mentioned procedure so as to select these terms that contain L^N and whose input weight and output weight fall within specific ranges. The selected terms compose the truncated IOWEF of the convolutional block code, with $w \in [1 \ldots N_W]$ and $d \in [1 \ldots N_D]$, for a code rate of 1/2 and a block of size N, where $N_W \le N$ and $N_D \le 2N$. The truncated IOWEF of a rate-1/3 PCCC can be computed by substituting the truncated IOWEF of the constituent convolutional block code into expressions (26)-(28).

6 Results

The proposed technique for the derivation of the IOWEF of a convolutional block code, described in the previous section, has been applied to RSC(1,7/5) and RSC(1,5/7) convolutional codes. The ML upper bound of the BEP performance of PCCCs employing them is obtained and compared with that presented in the literature. The simulation results obtained using iterative decoding and applying the BCJR decoding algorithm [11] after 8 iterations are plotted in Fig.4 against the corresponding theoretical bound.

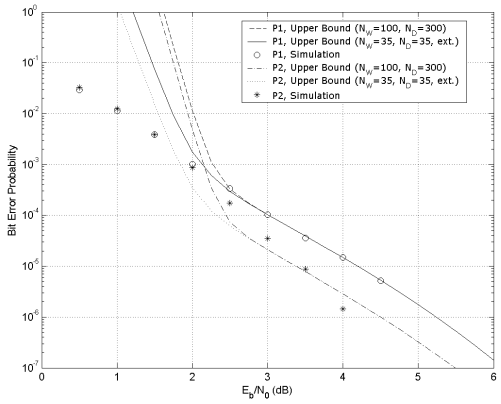

Fig. 4 Theoretical upper bounds and simulation of the BEP performance of $P1$ and $P2$ turbo codes.

For the results presented in Fig.4, both PCCCs use pseudorandom interleavers of size 100. The first turbo code, $P1$, employs RSC(1,7/5) as its constituent code whereas the second turbo code, $P2$, employs RSC(1,5/7). This comparison is also performed in [3] and [5] and the exact same ML upper bounds are derived.

The IOWEF of both $P1$ and $P2$ consists of terms whose input weight takes values up to N_W=100 and whose output weight takes values up to N_D=300. The computation of a truncated version of the IOWEF with N_W=35 and N_D=35 requires considerably less processing time compared to that of the full IOWEF. Although the truncated IOWEF enables the calculation of the first 35 terms only of the ML upper bound, we estimate the first 100 terms by means of extrapolation, as described in [4]. In Fig.4, it can be observed that the ML upper bound, which is derived based on the full IOWEF, does not result in additional accuracy.

7 Conclusions

In this paper we propose a new technique to evaluate the IOWEF of a convolutional block code. The technique involves the construction of an augmented modified state diagram of the constituent convolutional code, the derivation of the state equations and the evaluation of the transfer function of the convolutional block code.

The technique proposed by Divsalar *et al*. [3] requires the construction of the state diagram of the constituent convolutional code, the derivation of the state transition matrix and the computation of an inverse matrix, which is a function of the state transition matrix.

The approach presented by Benedetto and Montorsi [4] has two stages. In the first stage, an intermediate transfer function of the convolutional code is evaluated, yielding terms having particular input and output weights. In the second stage, the IOWEF of the convolutional block code is computed, since each term of the IOWEF associated with a specific input and output weight, can be expressed as a function of the relevant terms of the intermediate transfer function, i.e., those associated with identical input and identical output weights.

The technique we propose can be seen as a refinement of Benedetto's and Montorsi's approach. Owing to the introduction of the augmented state diagram, the IOWEF of the convolutional block code can be directly computed without the need of an intermediate transfer function. Furthermore, our approach can be easily extended to punctured turbo codes and enable us to accurately specify the best puncturing patterns, in terms of error rate performance [12].

References

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

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.

[3] D. Divsalar, S. Dolinar, R. J. McEliece and F. Pollara, "Transfer function bounds on the performance of turbo codes", JPL, Cal. Tech., TDA Progr. Rep. 42-121, Aug. 1995.

[4] S. Benedetto, G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes", *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 409-428, Mar. 1996.

[5] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes", *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591-600, May 1996.

[6] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design and iterative decoding", *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909-926, May 1998.

[7] W. E. Ryan, "Concatenated codes and iterative decoding" in *Wiley Encyclopedia of Telecommunications (J. G. Proakis, ed.)*, New York: Wiley and Sons, 2003, pp. 556-570.

[8] A. J. Viterbi, "Convolutional codes and their performance in communication systems" *Trans. Commun. Tech.*, vol. 19, no. 5, pp. 751-772, Oct. 1971.

[9] S. Benedetto, M. Mondin and G. Montorsi, "Performance evaluation of trellis-coded modulation schemes" *Proc. IEEE,* vol. 82, no. 6, pp. 833-855, June 1994.

[10] I. G. Bashmakova, *Diophantus and Diophantine Equations*, Washington DC: Math. Assoc. America, 1997.

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

[12] I. Chatzigeorgiou, M.R.D. Rodrigues, I. J. Wassell and R. Carrasco, "A novel technique for the evaluation of the transfer function of punctured turbo codes", in *Proc. Intl. Conf. Comm. (ICC'06)*, Istanbul, Turkey, July 2006 (to appear).