Revisiting the Calculation of the Effective Free Distance of Turbo Codes

Ioannis Chatzigeorgiou and Ian J. Wassell

Computer Laboratory, University of Cambridge
15 J.J. Thomson Avenue, Cambridge CB3 0FD, United Kingdom
E-mail: ic231@cam.ac.uk, ijw24@cam.ac.uk

Indexing Terms: Turbo Codes, Convolutional codes

We revisit the expression for the minimum Hamming weight of the output of a constituent convolutional encoder, when its input is a weight-2 sequence. The new expression particularly facilitates the calculation of the effective free distance of recently proposed schemes, namely non-systematic turbo codes and pseudo-randomly punctured turbo codes.

Introduction: Several authors [1, 2] have agreed that the performance of turbo codes [3] at the error floor region is largely determined by the weight-2 input minimum distance, which corresponds to the minimum Hamming weight among all codeword sequences generated by input sequences of weight two. If a turbo code $\mathcal{T}$ consists of *N* parallel concatenated convolutional codes separated by uniform interleavers, its weight-2 input minimum distance $d_2^{\mathcal{T}}$, which is also referred to as the *effective free distance* of $\mathcal{T}$, can be written as [4, 5]

$$d_2^{\mathcal{T}} = \sum_{k=1}^{N} d_2^{(k)} , \tag{1}$$

where $d_2^{(k)}$ is the weight-2 input minimum distance of the *k*-th constituent code.

Bounds on the weight-2 input minimum distance d_2 of a convolutional code as well as exact expressions are provided in [1, 4, 6]. Nevertheless, the exact expressions are accurate only when either the impulse response of the code is known [6] or the structure of the code meets particular criteria [1, 4]. Recently, Banerjee *et al.* [5] demonstrated that non-systematic turbo codes using quick-look-in (QLI) convolutional codes as constituent codes, can achieve lower error floors than those of conventional systematic turbo codes. Unfortunately QLI codes do not always meet the conditions of [1, 4], hence the corresponding expressions cannot be used to determine their weight-2 input minimum distances. In this Letter we relax the conditions of [1, 4] and we present expressions which allow the accurate calculation of d_2 for a wider set of convolutional codes.

Preliminaries: Let (r, 1, ν) represent a rate-1/r convolutional code of memory ν and $\mathbf{G}(D) = [\mathbf{P}^{(1)}(D)/\mathbf{Q}(D),...,\mathbf{P}^{(r)}(D)/\mathbf{Q}(D)]$ be the generator matrix of the recursive encoder for that code, where $\mathbf{P}^{(i)}(D) = p_\nu^{(i)}D^\nu + \ldots + p_1^{(i)}D + p_0^{(i)}$ denotes the i-th feed-forward generator polynomial and $\mathbf{Q}(D) = q_\nu D^\nu + \ldots + q_1 D + q_0$ corresponds to the feedback generator polynomial, with coefficients $p_j^{(i)}, q_j \in \{0,1\}$. Note than none of the feed-forward polynomials is equal to $\mathbf{Q}(D)$, whilst $\mathbf{P}^{(1)}(D)/\mathbf{Q}(D) = 1$ only if the convolutional code is systematic.

It was shown in [1, 4] that the weight-2 input minimum distance of a (r, 1, ν) recursive convolutional code is given by $d_2 = r(2 + 2^{\nu-1})$ if the code is non-systematic and $d_2 = 2 + (r-1)(2 + 2^{\nu-1})$ if the code is systematic. In both cases, it has been assumed that $\mathbf{Q}(D)$ is a primitive polynomial of order $\nu \geq 2$, i.e., $\deg \mathbf{Q}(D) = \nu$, while $\mathbf{P}^{(i)}(D)$ is a monic polynomial with constant term 1, i.e., $p_\nu^{(i)} = p_0^{(i)} = 1$. Consequently, $\deg \mathbf{P}^{(i)}(D) = \deg \mathbf{Q}(D) = \nu$.

Calculation of d_2 *when* $\deg \mathbf{P}^{(i)}(D) \leq \deg \mathbf{Q}(D)$: As previously, we assume that $\mathbf{Q}(D)$ is a primitive polynomial of order $\nu \geq 2$, since it has been shown that turbo codes using primitive feedback generator polynomials yield an excellent performance [1]. Let *u*(*t*) denote the input bit to the encoder at time step *t* and $r_m(t)$ represent the output of the *m*-th memory element, where *m*=1,…,*v*. Initially, we focus on the *i*-th non-systematic output of the encoder. The corresponding output bit $y^{(i)}(t)$ can be expressed as follows

$$
\begin{aligned}
y^{(i)}(t) = p_0^{(i)}u(t) \oplus (p_1^{(i)} \oplus q_1 p_0^{(i)})r_1(t) \oplus \ldots \\
\ldots \oplus (p_{\nu-1}^{(i)} \oplus q_{\nu-1} p_0^{(i)})r_{\nu-1}(t) \oplus (p_\nu^{(i)} \oplus p_0^{(i)})r_\nu(t),
\end{aligned}
\quad (2)
$$

where the symbol ⊕ denotes the mod-2 addition. We have also adopted the notation $d_{t_1 \to t_2}^{(i)}$ to represent the weight of the sequence generated by the *i*-th non-systematic output of the encoder during the transition from time step t_1 to time step t_2, i.e.,

$$d_{t_1 \to t_2}^{(i)} = \sum_{t=t_1}^{t_2-1} y^{(i)}(t). \quad (3)$$

If *L* is the period of the primitive feedback polynomial $\mathbf{Q}(D)$, the two nonzero bits of a weight-2 input sequence should be separated by *L*-1 zeroes such that the encoder returns to the zero state [7], i.e., $r_m(t) = 0$ for all *m*. Let *u*(0)=*u*(*L*)=1, while *u*(1)=…=*u*(*L*-1)=0. Note that the weight-2 input minimum distance of the *i*-th non-systematic output of the encoder is quantified by $d_{0 \to L+1}^{(i)}$. For convenience, we express $d_{0 \to L+1}^{(i)}$ as $d_{0 \to L+1}^{(i)} = d_{0 \to 1}^{(i)} + d_{1 \to L}^{(i)} + d_{L \to L+1}^{(i)}$ and compute each term separately:

- *t*: 0→1 – Assuming that the encoder was initialised to the zero state, we obtain $d_{0 \to 1}^{(i)} = y^{(i)}(0) = p_0^{(i)}$ from (2) and (3), since $u(0) = 1$ and $r_1(0) = \ldots = r_\nu(0) = 0$.
- *t*: 1→*L* – Let us first consider the case when *t*: 1→*L*+1 and u(*L*)=0. Owing to the properties of primitive polynomials, the output stream is a pseudo-noise sequence having weight $d_{1 \to L+1}^{(i)} = 2^{\nu-1}$, given that $\mathbf{P}^{(i)}(D) \neq \mathbf{Q}(D)$ [7]. Furthermore, when *t*=*L*, the encoder is in state 1 [7], i.e., $r_1(L) = \ldots = r_{\nu-1}(L) = 0$ and $r_\nu(L) = 1$. Hence, if *u*(*L*)=0 is the input bit, the encoder outputs

$y^{(i)}(L) = p_\nu^{(i)} \oplus p_0^{(i)}$, which is also the value of $d_{L\to L+1}^{(i)}$. However, an equivalent and more convenient form of the previous expression for the output weight is $d_{L\to L+1}^{(i)} = (p_\nu^{(i)} - p_0^{(i)})^2$. Consequently, we can compute the target quantity $d_{1\to L}^{(i)}$ by subtracting $d_{L\to L+1}^{(i)}$ from $d_{1\to L+1}^{(i)}$ and obtain $d_{1\to L}^{(i)} = 2^{\nu-1} - (p_\nu^{(i)} - p_0^{(i)})^2$, independently of the value of $u(L)$.

- *t*: $L \to L+1$ – We established that if $t=L$ then $r_\nu(L) = 1$, while the output of the remaining memory elements is zero. That is when the second nonzero bit, namely $u(L)=1$, of the weight-2 sequence is input to the encoder and forces it to return to the zero state. Using (2) and (3), we find that $d_{L\to L+1}^{(i)} = y^{(i)}(L) = p_\nu^{(i)}$.

Thus, the weight of the *i*-th non-systematic output sequence of the encoder for a weight-2 input sequence can be expressed as

$$\begin{aligned} d_{0\to L+1}^{(i)} &= d_{0\to 1}^{(i)} + d_{1\to L}^{(i)} + d_{L\to L+1}^{(i)} \\ &= p_0^{(i)} + 2^{\nu-1} - (p_\nu^{(i)} - 2p_0^{(i)}p_\nu^{(i)} + p_0^{(i)}) + p_\nu^{(i)} \\ &= 2^{\nu-1} + 2p_0^{(i)}p_\nu^{(i)}, \end{aligned} \tag{4}$$

using the fact that the value of a binary number, such as $p_j^{(i)}$, does not alter when it is raised to a power (e.g., $(p_j^{(i)})^2 = p_j^{(i)}$). The overall weight-2 input minimum distance of the rate-1/*r* recursive convolutional encoder can be obtained as follows

$$d_2 = \sum_{i=1}^{r} d_{0\to L+1}^{(i)} = \begin{cases} r2^{\nu-1} + 2\sum_{i=1}^{r} p_0^{(i)} p_\nu^{(i)}, & \text{if the code is non-systematic,} \\ 2 + (r-1)2^{\nu-1} + 2\sum_{i=2}^{r} p_0^{(i)} p_\nu^{(i)}, & \text{if the code is systematic.} \end{cases} \tag{5}$$

Extension to pseudo-randomly punctured codes: Pseudo-random (PR) puncturing, initially introduced in [7], is a method to increase the rate of a constituent recursive systematic convolutional code with generator matrix **G**(*D*)=[1, **P**(*D*)/**Q**(*D*)] from 1/2 to 1 by periodically eliminating particular bits from its output. Note that $\mathbf{Q}(D)$ should be primitive. It has been shown [8] that a rate-1/2 turbo code consisting of a rate-1 PR-punctured convolutional code and a rate-1 non-systematic convolutional code, yields a lower error floor than that of its rate-1/3 parent code. Following a similar reasoning as in the previous section, we can express (the proof has been omitted) the weight-2 input minimum distance of a PR-punctured convolutional code (1, 1, ν) as

$$d_2 = 2^{\nu-2} + 2p_0 p_\nu . \tag{6}$$

Conclusion: In this paper we expressed the weight-2 input minimum distance of a rate-1/*r* convolutional code as a function of the coefficients of its feed-forward generator polynomials $\mathbf{P}^{(i)}(D)$, with *i*=1,…,*r*, for a primitive feedback generator polynomial $\mathbf{Q}(D)$. This expression can be used to accurately compute the effective free distance of both conventional systematic turbo codes as well as non-systematic turbo codes [5, 8] that consist of convolutional codes with $\deg \mathbf{P}^{(i)}(D) \le \deg \mathbf{Q}(D)$.

Acknowledgment: This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/E012108/1.

References

1 BENEDETTO S. and MONTORSI G.: 'Design of parallel concatenated convolutional codes', *IEEE Trans.*, 1996, **COM-44**, pp. 591-600

2 PEREZ L. C., SEGHERS J. and COSTELLO D. J. Jr.: 'A distance spectrum interpretation of turbo codes', *IEEE Trans.*, 1996, **IT-42**, pp. 1698-1709

3 BERROU C., GLAVIEUX A. and THITIMAJSHIMA P.: 'Near Shannon limit error-correcting coding and decoding: Turbo-codes', Proc. ICC, Geneva, May 1993.

4 DIVSALAR D. and MCELIECE R. J.: 'The effective free distance of turbo codes', *Electron. Lett.*, 1996, **32**, (5), pp. 445-446

5 BANERJEE A., VATTA F., SCANAVINO B. and COSTELLO D. J. Jr.: 'Nonsystematic turbo codes', *IEEE Trans.*, 2005, **COM-53**, pp. 1841-1849

6 YOSCOVICH I. and SNYDERS J.: 'On the effective free distance of turbo codes', Proc. ITW, Killarney, June 1998.

7 CHATZIGEORGIOU I., RODRIGUES M. R. D, WASSELL I. J. and CARRASCO R.: 'A union bound approximation for rapid performance evaluation of punctured turbo codes', Proc. CISS, Baltimore, March 2007

8 CHATZIGEORGIOU I., RODRIGUES M. R. D., WASSELL I. J. and CARRASCO R.: 'Pseudo-random puncturing: A technique to lower the error floor of turbo codes', Proc. ISIT, Nice, June 2007