A Novel Method for Obtaining the Pattern of Low-Weight Codeword Components of Recursive Systematic Convolutional Codes

Bohulu Kwame Ackah and Chenggao Han
Graduate School of Informatics and Engineering,
The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182–8585, Japan
Email: {bohulu, han.ic}@uec.ac.jp

Abstract

In this paper, we present a novel low-complexity method for determining the pattern of the low-weight codeword components of any RSC code as well as the distance spectrum. Using our method, we list the partial distance spectrum for selected RSC codes up to a cut-off weight d_{max} and compare the simulation results to the bounds obtained via our novel method and the transfer function method.

I. Introduction

The *turbo code* (TC) [1], introduced by Claude Berrou in 1993 is one of the forward-error correcting codes that comes very close to satisfying the Shannon limit for AWGN channels. Due to its excellent performance, TCs have been used in many applications and adopted as the channel code for the LTE standard, IEEE 802.16 WiMAX (worldwide interoperability for microwave access) and DVB-RCS2 (2nd generation digital video broadcasting - return channel via satellite) standards [6].

The simplest and most common construction of a TC is to concatenate two *recursive systematic convolutional* (RSC) codes (usually of the same kind) parallely via an interleaver. One of the many reasons why the TC excels as a channel code is its ability to map low-weight parity-check sequences in the first component RSC code to high-weight parity-check sequences in the second component RSC code using an interleaver, which in turn generates TCs with a large minimum distance value.

The design of a good deterministic interleaver requires the complete knowledge of all the low-weight codeword component patterns in the RSC code and missing even one of these patterns can result in deterministic interleavers that generate TCs with sub-par error correction performance. The transfer function of an RSC code is an interleaver design tool that provides information about the different weights in the code, as well as their corresponding multiplicities (distance spectrum). However, it provides no information with regards to the pattern of the low-weight codeword components. As an added downside, the complexity of calculating the transfer function for a given RSC code increases with the number of states and other methods such as Mason's Rule [3] have to be used. To the best of our knowledge, there exists no interleaver design tool that provides knowledge of both the distance spectrum and the low-weight codeword component patterns. Because of this, many of the interleaver design methods end up completely ignoring certain important low-weight codewords. In [5] for example, the interleaver design method does not take into account the existence of low-weight codewords with systematic components of weight 3, especially for the $5/7$ RSC code, where such codewords are very dominant.

In this paper, we present a novel method that can be used to find the distance spectrum of an RSC code as well as the pattern of the low-weight codeword components. The complexity of our method is independent of the number of states of the RSC code and its ability to reveal low-weight codeword patterns of an RSC code makes it an excellent tool for use in interleaver design.

In order to validate our method, we generate a partial distance spectrum for specific RSC codes and compare it to the lower bound obtained via the transfer function method. We also compare the bounds obtained using our novel method to simulation results. In both cases, it is observed that the values begin to converge as E_b/N_0 increases.

The remainder of the research paper is organised as follows. Notations and definitions used in the research paper are introduced in Section II. In Section III, we discuss the distance spectrum and union bound of RSC codes and present the theory behind our novel method for obtaining the distance spectrum. Moving on to Section IV, we use our novel method to determine low-weight parity check patterns. Comparison of bounds obtained using our novel method to that obtained using the transfer function as well as simulation results are presented in Section VI and the paper concludes in Section VII.

II. Preliminaries

A polynomial in x, with degree M is an expression of the form

$$v(x)=\sum_{m=0}^{M} v_m x^m$$

where $v_m,\ 0 \leq m \leq M$, are called the *coefficients* and $v_m \neq 0$. If $v_M = 1,\ v(x)$ is called a *monic* polynomial. Moreover, the *Hamming weight* of $v(x)$, which is denoted by $w_H(v(x))$, is defined as the total number of non-zero coefficients. For two polynomials $v(x),\ \text{deg}(v(x)) = M$ and $w(x),\ \text{deg}(w(x)) = N$, the sum and product of $v(x)$ and $w(x)$ are defined as

$$v(x)+w(x)=\sum_{m=0}^{map\{M,N\}} (v_m + w_n)x^m$$

$$v(x)w(x)=\sum_{m=0}^{M+N}\sum_{i=0}^{m}(v_i w_{m-i})x^m$$

respectively.

For a prime number p, the Galois field with p elements, denoted as GF(p) and if the the set of integers $\{0, 1, p-1\}$ integers where addition and multiplication of 2 elements are carried out modulo-p. If the coefficients $v_m, 0 \leq m \leq M$ are elements of GF(p), $v(x)$ is called a polynomial over GF(p). A monic polynomial which cannot be factorised into lower degree polynomials over GF(p) is called a *prime polynomial*.

Let $v(x)$ be a prime polynomial with degree $M,\ M > 1$. Then a Galois field with p^M polynomial elements, represented by GF(p^M) can be constructed by considering the set of all polynomials with degree less than M in GF(p). GF(p^M) is called the *extension field* of GF(p) and GF(p) is called the *ground field* of GF(p^M).The addition operation in GF(p^M) is the same as regular addition, whiles the multiplication operation is carried out modulo-$v(x)$, where multiplication modulo-$v(x)$ means to divide the product of the 2 polynomials by $v(x)$ and return the remainder.

Elements in GF(p^M) can be represented by a power notation, *i.e.* $X^m,\ 0 \leq m \leq p^M - 1$, where X^2 may be used in place of $1+x$, for example. Through out this paper, the power notation will be used more often for the sake of convenience, with the appropriate conversion between the power and polynomial notation made known where necessary.

Let X be a non-zero element of GF(p^M).Then, ϵ denotes the *order* of X, and is defined as the least positive integer value such that $X^\epsilon = 1$, and X is called a *primitive element* if $\epsilon = p^M - 1$

. Let $v(x)$ be a prime polynomial with degree M. If $v(x)$ generates GF(p^M) such that X is a primitive element in GF(p^M), then $v(x)$ is called a *primitive polynomial*. Finally, the root of $v(x)$, denoted by β, is any element in GF(p^M) for which $v(x) = 0$.

III. Union Bound of RSC Codes

The outputs of an RSC code are determined by the input bit sequence $b(x)$, the states of the shift registers and the feedforward and feedback connections of shift registers that can be represented by a generator function.

As an instance, the generator function of a rate $1/2$ RSC code may be written as

$$\left[1 \; \frac{f(x)}{g(x)}\right]$$

where 1 yields the *systematic component* (SC) $b(x)$ while the *parity-check component* (PC) $h(x)$ is associated with the feedforward and feedback connections of the shift registers, specified by $f(x)$ and $g(x)$, respectively. The outputs $c(x)$ are a mixture of SC and PC as

$$c(x) = b(x^2) + xh(x^2) \tag{1}$$

where

$$h(x) = f(x)g^{-1}(x)b(x) \tag{2}$$

From (1), we have

$$w_H(c(x)) = w_H(b(x)) + w_H(h(x)) \tag{3}$$

and it is obvious that each low-weight codeword is a conjunction of two low-weight SC and PC.

Under the assumption of large frame sizes, the presence of $g^{-1}(x)$ in (2) may involve a particular sequence of bits that is repeated a large number of times, hence a high-weight PC. A low-weight PC occurs if and only if

$$b(x) \bmod g(x) \equiv 0 \tag{4}$$

and the inputs $b(x)$ which meet the condition in (4) is called a *return-to-zero* (RTZ) input. Thus, every RTZ input can be factorized by

$$b(x) = a(x)g(x) \tag{5}$$

Substituting (5) into (2), we can characterize the low-weight PC as

$$\begin{aligned} h(x) &= f(x) \cdot g^{-1}(x) \cdot a(x)g(x) \\ &= a(x)f(x) \end{aligned} \tag{6}$$

Finally, for a given RSC code, we can formulate our goal as, to find all $a(x)$s which satisfy (5) and (6) simultaneously. However, since there is essentially no difference between the two equations, in the next section, we present a method for determining the low-weight PC patterns for $2 \leq w_H(h(x)) \leq 3$

IV. The patterns of the low-weight PCs

To determine the details of the patterns of the low-weight PCs, we refer to (6) and we consider the solution of

$$h(x) \mod f(x) \equiv 0 \tag{7}$$

for $w_H(h(x)) = 2, 3$. To this end, we assume $f(x)$ can be factorized into K prime polynomials as

$$f(x) = \prod_{k=0}^{K-1} f_k^{\gamma_k}(x) \tag{8}$$

where $\gamma_0, \gamma_1, \cdots, \gamma_{K-1}$ are positive integers and we assume β_k is a root of $f_{\beta_k}(x)$ of order ϵ_k.

We first consider the simplest case $K = 1$, ***i.e.***, $f(x) = f_0^{\gamma_0}(x)$. Then, we can see from (6) that each root is also a root of $h(x)$. For the case $\gamma_0 = 1$, since all β_0^i, $0 \le i < \epsilon_0$, are distinct from each other, the condition

$$h(\beta_0^i) = 0, \quad 0 \le i < \epsilon_0 \tag{9}$$

is necessary and sufficient to fulfil (7). For $\gamma_0 > 1$, on the other hand, (9) is necessary but not sufficient for (7). For this case, although we may derive some solutions by differential equations

$$\left.\frac{d^{(j)}h(x)}{dx^j}\right|_{x=\beta_0^i} = 0, \quad 0 \le i < \epsilon_0,\ 1 \le j < \gamma_0 \tag{10}$$

we can not determine the patterns completely, since the operations on coefficients of the polynomial are done modulo-p. However, we can remove the ghost solutions **(perhaps has accurate name)** by careful confirmation.

By repeating the above discussion for the roots β_k, $0 < k < K$, and taking the intersection of the results, we can easily extend to the case $K > 1$.

A. *The patterns of the weight-2 PCs*

Each weight-2 PC can be written as

$$h(x) = 1 + x^a \tag{11}$$

without loss of generality. Thus, we have from (9) that

$$(\beta_0^i)^a = 1, \quad 0 \le i < \epsilon_0 \tag{12}$$

On the other hand, the order ϵ_0 is the least integer satisfying $\beta_0^{\epsilon_0} \equiv 1$, thus, a should satisfy the condition

$$a \bmod \epsilon_0 \equiv 0$$

Example 1. $f(x) = 1 + x + x^2$.
For this case, since $x^1 = x$, $x^2 \equiv 1+x$, and $x^3 \equiv 1 \bmod f(x)$, the order of the root β_0 is $\epsilon_0 = 3$ and a should be a multiple of 3. The corresponding values for $a(x)$ and $h(x)$ are shown in Table I for the first four valid values of a.

TABLE I: $f(x) = 1 + x + x^2$

$a(x)$	$h(x)$
$1+x$	$1+x^3$
$1+x+x^3+x^4$	$1+x^6$
$1+x+x^3+x^4+x^6+x^7$	$1+x^9$
$1+x+x^3+x^4+x^6+x^7+x^9+x^{10}$	$1+x^{12}$

We may write the weight-2 PCs in general form as $h(x) = 1+x^{3\ell},\ \ell > 1$ and the corresponding $a(x)$ is given by

$$a(x) = \sum_{\ell=0}^{L-1} x^{3\ell}(1+x)$$

Example 2. $f(x) = 1 + x + x^2 + x^3 + x^4$
We can confirm that the order of β_0 is $\epsilon_0 = 5$. This means that a should be a multiple of 5. The corresponding values for $a(x)$ and $h(x)$ are shown in Table II with general forms for $\ell > 1$

TABLE II: $f(x) = 1 + x + x^2 + x^3 + x^4$

$a(x) = \sum_{\ell=0}^{L-1} x^{5\ell}(1+x)$	$h(x) = 1 + x^{5\ell}$
$1+x$	$1+x^5$
$1+x+x^5+x^6$	$1+x^{10}$
$1+x+x^5+x^6+x^{10}+x^{11}$	$1+x^{15}$
$1+x+x^5+x^6+x^{10}+x^{11}+x^{15}+x^{16}$	$1+x^{20}$

Example 3. $f(x) = 1 + x^2$

Since

$$f(x) = (1 + x)^2$$

and the order of the root $\beta_0 = 1$ is $\epsilon_0 = 1$, we obtain from (9) and (10)

$$(\beta_0)^a = 1 \tag{13}$$

$$a(\beta_0)^{(a-1)} = 0 \tag{14}$$

Although (13) indicates a can be any positive integer, we can see from (14) that a should be even number. The corresponding values for $a(x)$ and $h(x)$ are shown in Table III with general forms for $\ell > 1$.

TABLE III: $f(x) = 1 + x^2$

$a(x) = \sum_{\ell=0}^{L-1} x^{2\ell}$	$h(x) = 1 + x^{2\ell}$
1	$1 + x^2$
$1 + x^2$	$1 + x^4$
$1 + x^2 + x^4$	$1 + x^6$
$1 + x^2 + x^4 + x^6$	$1 + x^8$

Example 4. $f(x) = 1 + x^2 + x^3 + x^4$

$f(x)$ can be written as

$$f(x) = \prod_{k=0}^{1} f_k(x)$$

where

$$f_0(x) = 1 + x,\ \ f_1(x) = 1 + x + x^3$$

For $f_0(x), x \equiv 1$, and $x^1 \equiv 1 \bmod f_0(x)$, which means the order of the root β_0 is $\epsilon_0 = 1$ and a_0 should be a multiple of 1. Again, for $f_1(x), x^3 \equiv 1 + x$, and $x^7 \equiv 1 \bmod f_1(x)$, which means the order of the root β_1 is $\epsilon_1 = 7$ and a_1 should be a multiple of 7. Finally, valid values of a should be a multiple of the least common multiples of a_0 and a_1, which means a should be a

TABLE IV: $f(x) = 1 + x^2 + x^3 + x^4$

$a(x) = \sum_{\ell=0}^{L-1} x^{7\ell}(1 + x^2 + x^3)$	$h(x) = 1 + x^{7\ell}$
$1 + x^2 + x^3$	$1 + x^7$
$1 + x^2 + x^3 + x^7 + x^9 + x^{10}$	$1 + x^{14}$
$1 + x^2 + x^3 + x^7 + x^9 + x^{10} + x^{14} + x^{16} + x^{17}$	$1 + x^{21}$

multiple of 7. The corresponding values for $a(x)$ and $h(x)$ are shown in Table IV with general forms for $\ell > 1$.

Example 5. $f(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$
$f(x)$ can be written as

$$f(x) = \prod_{k=0}^{1} f_k(x)$$

where

$$f_0(x) = 1 + x^2 + x^3, \;\; f_1(x) = 1 + x + x^3$$

For $f_0(x), x^3 \equiv x^2 + 1$, and $x^7 \equiv 1 \bmod f_0(x)$, which means the order of the root β_0 is $\epsilon_0 = 7$ and a_0 should be a multiple of 7. Again, for $f_1(x), x^3 \equiv 1 + x$, and $x^7 \equiv 1 \bmod f_1(x)$, which means the order of the root β_1 is $\epsilon_1 = 7$ and a_1 should be a multiple of 7. Finally, valid values of a should be a multiple of the least common multiples of a_0 and a_1, which means a should be a multiple of 7. The corresponding values for $a(x)$ and $h(x)$ are shown in Table V with general forms for $\ell > 1$.

TABLE V: $f(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$

$a(x) = \sum_{\ell=0}^{L-1} x^{7\ell}(1 + x)$	$h(x) = 1 + x^{7\ell}$
$1 + x$	$1 + x^7$
$1 + x + x^7 + x^8$	$1 + x^{14}$
$1 + x + x^7 + x^8 + x^{14} + x^{15}$	$1 + x^{21}$

Example 6. $f(x) = 1 + x^2 + x^3 + x^4 + x^6$
$f(x)$ can be written as

$$f(x) = \prod_{k=0}^{1} f_k(x)$$

where

$$f_0(x) = 1 + x + x^2,\ f_1(x) = 1 + x + x^2 + x^3 + x^4$$

From Example IV-B and Example 2, we know that $a_0 = 3$ and $a_1 = 5$. Hence, valid values of a should be a multiple of the least common multiples of a_0 and a_1, which means a should be a multiple of 15. The corresponding values for $a(x)$ and $h(x)$ are shown in Table VI with general forms for $\ell > 1$.

TABLE VI: $f(x) = 1 + x^2 + x^3 + x^4 + x^6$

$a(x) = \sum_{\ell=0}^{L-1} x^{15\ell}(1 + x^2 + x^3 + x^6 + x^7 + x^9)$	$h(x) = 1 + x^{15\ell}$
$1 + x^2 + x^3 + x^6 + x^7 + x^9$	$1 + x^{15}$
$1 + x^2 + x^3 + x^6 + x^7 + x^9 + x^{15} + x^{17} + x^{18} + x^{21} + x^{22} + x^{24}$	$1 + x^{30}$

B. The patterns of the weight-3 PCs

Each weight-3 PC can be written as

$$h(x) = 1 + x^a + x^b,\ a \neq b \tag{15}$$

without loss of generality. Fixing β_0 into $h(x)$ we have

$$\beta_0^a + \beta_0^b = 1 \tag{16}$$

To solve (16), we refer to the table of the extended field for GF(2^M), and we can find the valid $(\eta,\ \zeta)$ pairs $\text{ s.t. } X^\eta + X^\zeta = 1$. If there are no valid $(\eta,\ \zeta)$ pairs, then there is no parity check component of weight 3 for the given $f(x)$. Depending on the Galois field, there might be multiple values of $(\eta,\ \zeta)$ that satisfy (16), so we represent the set of all $(\eta,\ \zeta)$ pairs by $\mathcal{Z}$. Then, any valid value $(a,\ b)$ values should satisfy the condition

$$(a, b) \equiv (\eta,\ \zeta) \bmod \epsilon_0,\ (\eta,\ \zeta) \in \mathcal{Z} \tag{17}$$

Example 7. $f(x) = 1 + x + x^2$

The elements of GF(2^2) are shown in Table VII and it is obvious that $\mathcal{Z} = \{(1, 2)\}$. This means that $(a, b) = (3\ell + 1,\ 3n + 2),\ l = n = \{0, 1, ...\}$. The corresponding values for $a(x)$ and $h(x)$ are shown in the table below for the first four valid values of (a, b).

TABLE VII: Non-zero Elements of GF(2^2) generated by $f(x) = 1 + x + x^2$

power representation	actual value
$X^0 = X^3 = 1$	1
X	x
X^2	$1 + x$

TABLE VIII: $f(x) = 1 + x + x^2$

$a(x)$	$h(x)$
1	$1 + x + x^2$
$1 + x + x^2$	$1 + x^2 + x^4$
$1 + x + x^3$	$1 + x^4 + x^5$
$1 + x^2 + x^3$	$1 + x + x^5$

We may write the weight-3 PCs in general form as $h(x) = 1 + x^{3\ell+1} + x^{3n+2},\ \ell,\ n \geq 0$

a) Case2: $f(x)$ can be factorised into multiple irreducible polynomials.

For this case, we can write $f(x)$ as

$$f(x) = \prod_{k=1}^{K} f_k(x)$$

where $f_k(x)$ is an irreducible polynomial of order M_k with root $\beta \in$ GF(2^{M_k}), $\beta^{\epsilon_k} = 1$. For each $f_k(x)$, we refer to the table of the extended field it generates and form the set $\mathcal{Z}_k$, which contains all the valid $(\eta^{(k)},\ \zeta^{(k)})$ pairs for that particular $f_k(x)$. If that set exists, then, for that $f_k(x)$ the following condition is met

$$\mathcal{AB}_k = \{(a_k,\ b_k) \mid (a_k,\ b_k) \equiv (\eta^{(k)},\ \zeta^{(k)}) \bmod \epsilon_k,\ (\eta^{(k)},\ \zeta^{(k)}) \in \mathcal{Z}_k\} \tag{18}$$

and

$$(a,\ b) \in \bigcup_{k=1}^{K} \mathcal{AB}_k \tag{19}$$

For the special case where $f(x)$ can be factorised into equal irreducible polynomial, the above condition simplifies to

$$(a,\ b) \equiv (K\eta,\ K\zeta) \bmod \epsilon,\ (\eta,\ \zeta) \in \mathcal{Z}$$

Example 8. $f(x) = 1 + x^2$
$f(x)$ can be written as $(1 + x)^2$. $(1 + x)$ is a primitive polynomial for GF(2). The elements in GF(2) are 1 and β. In this field, there are no valid (e, f) pair values; therefore, $h(x)$ such that $w_H(h(x)) = 3$ does not exist for $f(x) = 1 + x^2$.

V. Codeword Component Pattern List and the Union Bound

In this section, we outline our method for obtaining the list of all low weight codewords for a given RSC code.

Let $\mathcal{A}_h(d)$ be the set of all $a(x)$ which yields weight-d parity-check component *i.e.*, $w_H(h(x)) = w_H(a(x)f(x)) = d$ for $a(x) \in \mathcal{A}_h(d)$. Similarly $\mathcal{A}_b(d)$ is the set of all $a(x)$ which yields weight-d systematic component *i.e.*, $w_H(b(x)) = w_H(a(x)g(x)) = d$ for $a(x) \in \mathcal{A}_b(d)$ and $\mathcal{A}_c(d)$ is the set of all $a(x)$ which yields weight-d codeword *i.e.*, $w_H(c(x)) = w_H(a(x)f(x)) + w_H(a(x)g(x)) = d$ for $a(x) \in \mathcal{A}_c(d)$.

From (3), when $w_H(b(x)), w_H(h(x)) \geq 2$, we have

$$\mathcal{A}_c(d) = \bigcup_{\ell=2}^{d-2} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \tag{20}$$

However, to determine $\mathcal{A}_b(\ell)$ or $\mathcal{A}_h(\ell)$ for a large ℓ is a complex task in general. Thus, in this paper, we replace the set $\mathcal{A}_c(d)$ by the approximated set $\mathcal{A}'_c(d)$ as defined in (21)

$$\begin{aligned} \mathcal{A}_c(d) \approx \mathcal{A}'_c(d) = &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \right\} \bigcup \\ &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(d-\ell) \cap \mathcal{A}_h(\ell)\} \right\} \end{aligned} \tag{21}$$

where some codewords in $\mathcal{A}_c(d)$ with $\ell \approx d-\ell$ may be ignored in $\mathcal{A}'_c(d)$.

Example 9. If we set $d = 8$ and $\alpha = 1$, $\mathcal{A}'_c(8)$ becomes

$$\begin{aligned} \mathcal{A}'_c(8) = &\Big\{ \{\mathcal{A}_b(2) \cap \mathcal{A}_h(6)\} \bigcup \{\mathcal{A}_b(3) \cap \mathcal{A}_h(5)\} \Big\} \bigcup \\ &\Big\{ \{\mathcal{A}_b(6) \cap \mathcal{A}_h(2)\} \bigcup \{\mathcal{A}_b(5) \cap \mathcal{A}_h(3)\} \Big\} \end{aligned}$$

We can see that $\{\mathcal{A}_b(4) \cap \mathcal{A}_h(4)\}$ is not used in $\mathcal{A}'_c(8)$, event though it is used in $\mathcal{A}_c(8)$.

Once we obtain the set

$$\bigcup_{d=d_{\text{min}}}^{d_{\text{max}}} \mathcal{A}'_c(d)$$

, we can list the low weight codeword component patterns for a given RSC code using (5) and (6). We set $d_{\text{max}} = d_{\text{min}} + 3$ and list the low weight codeword component patterns for the $5/7,\ 37/21$ and $23/35$ RSC codes in Tables IX, X and XI respectively.

TABLE IX: Partial Codeword Component Pattern Distance Spectrum for the $5/7$ RSC code, $d_{\text{free}} = 5$

$a(x)$	$b(x)$	$h(x)$
1	$1+x+x^2$	$1+x^2$
$1+x^2$	$1+x+x^3+x^4$	$1+x^4$
$1+x$	$1+x^3$	$1+x+x^2+x^3$
$1+x^2+x^4$	$1+x+x^3+x^5+x^6$	$1+x^6$
$1+x^2+x^3$	$1+x+x^5$	$1+x^3+x^4+x^5$
$1+x+x^2$	$1+x^2+x^4$	$1+x+x^3+x^4$
$1+x+x^3$	$1+x^4+x^5$	$1+x+x^2+x^5$
$1+x^2+x^4+x^6$	$1+x+x^3+x^5+x^7+x^8$	$1+x^8$
$1+x+x^3+x^4$	$1+x^6$	$1+x+x^2+x^4+x^5+x^6$

TABLE X: Partial Codeword Component Pattern Distance Spectrum for the $37/21$ RSC code, $d_{\text{free}} = 6$

$a(x)$	$b(x)$	$h(x)$
$1+x$	$1+x+x^4+x^5$	$1+x^5$
1	$1+x^4$	$1+x+x^2+x^3+x^4$
$1+x+x^5+x^6$	$1+x+x^4+x^6+x^9+x^{10}$	$1+x^{10}$

In order to validate our novel method, we can calculate the lower bound for each RSC code by modifying the union bound of the bit-error rate equation in [4], which gives us

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\max}} \sum_{a(x)\in\mathcal{A}'_c(d)} w_H(a(x)g(x)) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{22}$$

TABLE XI: Partial Codeword Component Pattern Distance Spectrum for the $23/35$ RSC code, $d_{\text{free}} = 7$

$a(x)$	$b(x)$	$h(x)$
$1+x^2+x^3$	$1+x^7$	$1+x+x^2+x^6+x^7$
1	$1+x^2+x^3+x^4$	$1+x+x^4$
$1+x+x^2+x^3+x^5$	$1+x^3+x^4+x^8+x^9$	$1+x^7+x^9$
$1+x+x^2+x^3+x^5+x^7+x^8$	$1+x+x^3+x^4+x^7+x^{12}$	$1+x^{11}+x^{12}$
$1+x^2+x^3+x^7+x^9+x^{10}$	$1+x^{14}$	$1+x+x^2+x^6+x^8+x^9+x^{13}+x^{14}$

VI. Results

In this section, we compare the bounds obtained via our novel method to bounds obtained using the transfer function method as well as the simulation results for three different RSC codes, each with a frame size $N = 64$.

We set $d_{\text{max}} = d_{\text{min}} + 3$, and calculate the lower bounds for our novel method and the transfer function using (22) and its original version in [4] respectively.

Fig. 1: Old Bound vs New Bound vs Simulation for 5/7 RSC Code

Fig. 1 shows the simulation results for the $5/7$ RSC code as well as the lower bounds obtained using the transfer function as well as our novel method. The feedforward connection has the polynomial representation $1+x^2$ and can be factorized into the irreducible polynomial $1+x$. From Example 8, we observe that there are no low-weight codewords with parity-check components of weight 3. The feedback connection has the polynomial representation $1+x+x^2$, which is an irreducible polynomial and from Example 2 and Example 7, we can confirm that the low-weight codewords have systematic components with weight 2 and weight 3. In Fig. 1, we observe that there is some difference between the new (novel method) bound and the old (transfer function) bound, but they tend to converge as E_b/N_0 increases. This suggests that the approximation $\mathcal{A}'_c(d_{\text{max}})$ used in our novel method is sufficient for the $5/7$ RSC code.

Fig. 2: Old Bound vs New Bound vs Simulation for 37/21 RSC Code

Fig. 2 shows the simulation results for the $37/21$ RSC code as well as the lower bounds obtained using the transfer function as well as our novel method. The feedforward connection has the polynomial representation $1+x+x^2+x^3+x^4$, which is an irreducible polynomial. We observe from Example 3 that low-weight codewords with weight 2 parity check components exist. However, there are no low-weight codeword with parity check components of weight 3, as seen in Table X. The feedback connection has the polynomial representation $1+x^4$, and from Example 8, we can deduce that low-codewords with systematic components of weight 3 do not exist. In Fig. 2, we observe that there is some difference between the new (novel method) bound and the old (transfer function) bound, but they tend to converge as E_b/N_0 increases. This suggests that the approximation $\mathcal{A}'_c(d_{\text{max}})$ used in our novel method is also sufficient for the $37/21$ RSC code.

Fig. 3 shows the simulation results for the $23/35$ RSC code as well as the lower bounds obtained using the transfer function as well as our novel method. The feedforward connection

Fig. 3: Old Bound vs New Bound vs Simulation for 23/35 RSC Code

has the polynomial representation $1+x+x^4$, which is an irreducible polynomial. The low-weight codeword has parity check components of weight 2 and weight 3. However, since parity-check components yield high weight codewords, there are not included in our approximation of the lower bound, as can be observed from Table X. The feedback connection has the polynomial representation $1 + x^2 + x^3 + x^4$, which can be factorized into 2 irreducible polynomials and there are no low-codewords with systematic components of weight 3 because $1 + x$ is a factor. In Fig. 3, we observe that the old (transfer function) bounds and simulation results converge as the E_b/N_0 value increases. However, there is some difference between the new (novel method) bound and the old (transfer function) bound, even as E_b/N_0 increases. This suggests that the approximation used in our novel method is insufficient for this $23/35$ RSC code and considering $w_H(h(x)),\ w_H(b(x)) = 4$ might yield a more accurate bound.

VII. Conclusion

In this paper, we presented a method for obtaining the systematic and parity check component patterns of an RSC codeword, given the RSC code and a cut-off weight, d_{max}. Compared to the transfer function method, it has low complexity and the knowledge of the pattern of the systematic and parity check components makes it a very useful for interleaver design. To validate our method, we compared the lower-bound obtained using our novel method with the lower-bound obtained via the transfer function as well as the simulation results for three RSC codes. Results show that whiles our method is sufficient for most RSC codes, considering codeword components with $w_H(b(x)), w_H(h(x)) = 4$ in our approximation, might yield more accurate BER bounds.

REFERENCES

[1] C. Berrou, A. Glavieux and P. Thitimajshima, ”Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1,” Proceedings of ICC ’93 - IEEE International Conference on Communications, Geneva, Switzerland, 1993, pp. 1064-1070 vol.2, doi: 10.1109/ICC.1993.397441.

[2] John G. Proakis, Masoud Salehi. “Digital Communications”, Fifth Edition,Chapter 8, McGraw-Hill.

[3] Todd K. Moon. “Error Correcting Codes”,Chapter 12, John Wiley & Sons.

[4] Alain Glavieux, “Channel Coding in Communication Networks: From Theory to Turbocodes”, Chapter 3, John Wiley & Son.

[5] Jing Sun and O. Y. Takeshita, ”Interleavers for turbo codes using permutation polynomials over integer rings,” in IEEE Transactions on Information Theory, vol. 51, no. 1, pp. 101-119, Jan. 2005, doi: 10.1109/TIT.2004.839478.

[6] R. Garzn-Bohrquez, C. Abdel Nour and C. Douillard, ”Protograph-Based Interleavers for Punctured Turbo Codes,” in IEEE Transactions on Communications, vol. 66, no. 5, pp. 1833-1844, May 2018, doi: 10.1109/TCOMM.2017.2783971.