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 [7].

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 and low multiplicity.

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. 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 briefly review the RSC codes. Moving on to Section IV, we present the theory behind our novel method and use it to generate a partial distance spectrum using a union bound-like approach for selected RSC codes in V. 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

Definition 1. Polynomials over GF(p)

A polynomial $v(x)$ over GF(p) is defined as

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

where $v_m \in$ GF(p), $v_M \neq 0$. p is a prime number and M is the degree of $v(x)$. Because we are working in the binary domain, we set $p = 2$.

Definition 2. Hamming weight of $v(x),\ w_H(v(x))$

Let $v(x)$ be a polynomial in GF(2), then the Hamming weight of $v(x)$, represented by $w_H(v(x))$ is equal to the number of terms with non-zero coefficients.

Definition 3. Prime Polynomial

If $v_M = 1$, then $v(x)$ is a *monic* polynomial. If $v(x)$ cannot be factorised into lower degree poynomials over GF(2), it is an *irreducible* polynomial. If $v(x)$ is both monic an irreducible, it is a *prime* polynomial.

Definition 4. Order of an Element in GF(2^m)

Let β^m represent a non-zero element in GF(2^M, $1 \leq m \leq 2^M - 1$). The least positive integer value ϵ such that $(\beta^m)^\epsilon = 1$ is known as the *order* of β^m

Definition 5. Primitive Element

Let β^m represent a non-zero element in GF(2^M). If its order ϵ is such that $\epsilon = 2^M - 1$, then it is a primitive element.

Definition 6. Primitive Polynomial

Let $v(x)$ be a prime polynomial with degree M. If GF(2^M) is constructed based on $v(x)$ and $\beta^1 = \beta$ is a primitive element, then $v(x)$ is a primitive polynomial. Alternately, if $v(x)$ does not divide $1 + x^\epsilon$ for any $\epsilon < 2^M - 1$, then it is a primitive polynomial.

Definition 7. $(e,\ f) \bmod 2^M - 1$

Let $(e\ f)$ represent a pair of non-zero positive integers. Then $(e,\ f) \bmod 2^M - 1$ is shorthand for the operation $(e \bmod 2^M - 1,\ f \bmod 2^M - 1)$.

III. A BRIEF REVIEW OF RSC CODES

The output bits of an RSC code are generated using the feedforward and feedback connections of shift registers determined by its generator function. The generator function may be written in polynomial notation as $\left[1 \ \frac{f(x)}{g(x)}\right]$, where 1 yields the systematic (input) bits of the output while the parity check bits of the output are specified by $f(x)$ and $g(x)$ that represent the feedforward and feedback connections of the shift registers respectively.

For a given RSC code, the distance spectrum provides information concerning the multiplicity of a codeword for a fixed weight and it is an effective tool to evaluate its error-correcting capability. Since higher-weight codewords have very little effect on its overall error-correcting capability, it is not unusual to use a partial distance spectrum, where the largest codeword weight value is set to d_{max}.

The distance spectrum of the RSC code can be obtained from its transfer function, denoted by

$$T(Y, X) = \sum_{d=0}^{\infty} \sum_{w=0}^{\infty} a(d, w) Y^d X^w$$

where $a(d, w)$ is the number of codewords of weight d generated by an input message of weight w. The transfer function enumerates all the paths that diverge from and then return to the initial state [3], *i.e.* the *return-to-zero* (RTZ) inputs. The complexity involved in deriving the transfer function increases as the number of states of the RSC code increases and other methods such as Mason's Rule [3] have to be used.

IV. Novel Method to Determine the Low-Weight Parity-Check Pattern

In this section, we present our novel method for obtaining what we have named the *codeword component pattern distance spectrum*. Our novel method can be seen as the combination of two different but related methods. The first method is quite simple and makes use of the fact that in the polynomial domain, systematic components that are RTZ inputs and their corresponding parity-check components share a common factor. The second method shows how to obtain this common factor when the weight of the parity check component is fixed for a given RSC code. Through out this section, $c(x),\ b(x)$ and $h(x)$ represesent the RSC codeword, the systematic component of the codeword and the parity check component of the codeword, respectively in polynomial notation.

A. The Characteristics of Low-weight Codewords

Since each RSC codeword is made up of two codeword components $b(x)$ and $h(x)$, it is obvious that the weight of the codeword $c(x)$ is given by

$$w_H(c(x)) = w_H(b(x)) + w_H(h(x)) \tag{1}$$

We first consider the parity check component, which can be expressed as

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

If we consider large frame sizes, the presence of $g^{-1}(x)$ means that within $h(x)$ is a particular sequence of bits that is repeated a large number of times. This results in a large parity weight, and by extension, a relatively high-weight codeword. The only time this is not the case is when

$$b(x) \bmod g(x) \equiv 0 \tag{3}$$

This results in a relatively low-weight parity bit sequence, which might produce a low-weight codeword. Any $b(x)$ that meets the condition in (3) can be written as

$$b(x) = a(x)g(x) \tag{4}$$

where $a(x)$ is a monic polynomial with $a_0 = 1$. By fixing $b(x)$ from (4) into (2), we have

$$\begin{aligned} h(x) &= f(x) \cdot g^{-1}(x) \cdot a(x)g(x) \\ &= a(x)f(x) \end{aligned} \tag{5}$$

Thus, for a given $f(x)$ and $g(x)$, our goal is to find all $a(x)$s which generate low-weight codewords components in (4) and (5) simultaneously. However, since there is essentially no difference between the general structure of $h(x)$ and $b(x)$, we restrict our our attention to the low-weight parity check patterns in (5) and in the following, we present a method for determining valid values of $h(x)$ when $2 \leq w_H(h(x)) \leq 3$.

B. The Characteristic of Weight 2 Parity Check Pattern

For this weight case, we can write $h(x)$ as $h(x) = 1 + x^a$ without any loss of generality. Let M be the degree of $f(x)$ and $\alpha_m,\ 0 \leq m \leq M-1$, be the roots of $f(x)$ satisfying $f(\alpha_m) = 0$ for all $0 \leq m \leq M-1$. Then from (5), we have

$$\begin{aligned} h(\alpha_m) &= 0 \\ 1 + (\alpha_m)^a &= 0 \end{aligned} \tag{6}$$

for all $0 \leq m \leq M-1$ and therefore, we can reformulate our goal as to find low-weight polynomials $h(x)$ which satisfy (6) for all roots of $f(x)$.

Now we consider the characteristics of the roots of $f(x)$ for the folowing 2 cases.

a) Case1: $f(x)$ is a single irreducible polynomial

For this case, $f(x)$ has M distinct roots and each root α_m can be expressed as

$$\alpha_m = \beta^{2^m},\ 0 \leq m \leq M-1 \tag{7}$$

where β^{2^m} is primitive element in GF(2^M) with order $\epsilon,\ \epsilon | 2^M - 1$. Now, if we consider the possible values of $a > 0$ such that

$$(\beta^{2^m})^a = 1$$

then,

$$a \bmod \epsilon \equiv 0$$

since $(\beta)^\epsilon = 1$

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

$f(x)$ generates the field GF(2^2). There are 2 distinct roots of $f(x)$, β and β^2. The order of β and β^2 is $\epsilon = 3$, and the valid values of a are $a = \{3, 6, 9, \cdots\}$. The corresponding values for $a(x)$ and $h(x)$ are shown in Table I for the first four valid values of a.

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

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

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

$f(x)$ can be used used to generate GF(2^4). There are 4 distinct roots and $\beta,\ \beta^2,\ \beta^3$ and β^4. The order of each root is $\epsilon = 5$ and therefore, the valid values of a are $a = \{5, 10, 15, \cdots\}$. The corresponding values for $a(x)$ and $h(x)$ are shown in Table II for the first four valid values of a.

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

$a(x)$	$h(x)$
$1 + x$	$1 + x^5$
$1 + x + x^5 + x^6$	$1 + x^{10}$
$1 + x + x^5 + x^6 + x^{10} + x^{11}$	$1 + x^{15}$
$1 + x + x^5 + x^6 + x^{10} + x^{11} + x^{15} + x^{16}$	$1 + x^{20}$

b) 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 with order ϵ_k. For each $f_k(x)$, the valid values of a_k are such that

$$a_k \bmod \epsilon_k \equiv 0$$

and the valid values of a are such that

$$a \in \bigcap_{k=1}^{K} a_k$$

This means that a satisfies the condition

$$a \bmod \prod_{k=1}^{K} \epsilon_k \equiv 0$$

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

$$a \bmod \epsilon K \equiv 0$$

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

$f(x)$ can be written as

$$f(x) = (1 + x)^2,\ K = 2$$

$1 + x$ is prime in $GF(2)$ and has β as its root. The order of β is 1. Since $f(x)$ is made up of equal repeated polynomial and $K = 2$, the valid values of $a = \{2, 4, 6, \cdots\}$. The corresponding values for $a(x)$ and $h(x)$ are shown in Table III for the first four valid values of a.

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

$a(x)$	$h(x)$
1	$1 + x^2$
$1 + x^2$	$1 + x^4$
$1 + x^2 + x^4$	$1 + x^6$
$1 + x^2 + x^4 + x^6$	$1 + x^8$

C. The Characteristic of Weight 3 *Parity Check Pattern*

For this weight case,

$$h(x) = 1 + x^a + x^b,\ a \neq b \tag{8}$$

without loss of generality. We consider the characteristics of the roots of $f(x)$ for the folowing 2 cases.

a) Case1: $f(x)$ is a single irreducible polynomial

If $f(x)$ is an irreducible polynomial, then it can be used to generate the extended field GF(2^M). The non-zero elements of the extended field are represented by $\beta^m,\ 1 \leq m \leq 2^M - 1$. Also, from the discussion in the previous section, we know that $f(x)$ has M distinct roots, one of them being β. Substituting β into (8), we get

$$\begin{aligned} h(\beta) &= 0 \\ 1 + \beta^a + \beta^b &= 0 \end{aligned} \tag{9}$$

We can then reformulate our task as to find all β^m such that

$$\beta^{\eta} + \beta^{\zeta} = 1,\ \eta \neq \zeta \tag{10}$$

By refering to the table of the extended field for GF(2^m), we can find the valid $(\eta,\ \zeta)$ pairs s.t. $\beta^{\eta} + \beta^{\zeta} = 1$. If there are no valid $(\eta,\ \zeta)$ pairs, then there is no parity check component of weight 3 for the given $f(x)$. We represent the set of $(\eta,\ \zeta)$ pairs as $\boldsymbol{z} = \{(\eta_1,\ \zeta_1), (\eta_2,\ \zeta_2), \cdots\}$. Then, any valid value $(a,\ b)$ values should satisfy the condition

$$(a, b) \equiv (\eta,\ \zeta) \bmod \epsilon,\ (\eta,\ \zeta) \in \boldsymbol{z} \tag{11}$$

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

The elements of GF(2^2) are shown in Table IV and it is obvious that there is exactly 1 valid (η, ζ) pair s.t. $\beta^{\eta} + \beta^{\zeta} = 1$ and that is the pair $(1, 2)$. This means that valid values of the (a, b) pairs are any values s.t. $(a, b) \equiv (1, 2) \bmod 3$. 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 IV: Non-zero Elements of GF(2^2) generated by $f(x) = 1 + x + x^2$

power representation	actual value
$\beta^0 = \beta^3 = 1$	1
β	β
β^2	$1 + \beta$

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

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

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

We may write $f(x)$ as

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

where $f_k(x)$ is an irreducible polynomial with order ϵ_k For each $f_k(x)$, we refer to the table of the extended field it generates and form the set $\boldsymbol{z}_k$, 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

$$(a_k,\ b_k) \equiv (\eta^{(k)},\ \zeta^{(k)}) \bmod \epsilon_k,\ (\eta^{(k)},\ \zeta^{(k)}) \in \boldsymbol{z}_k \tag{12}$$

and

$$(a,\ b) \equiv \bigcup_{k=1}^{K} (\eta^{(k)},\ \zeta^{(k)}) \bmod \epsilon_k,\ (\eta^{(k)},\ \zeta^{(k)}) \in \boldsymbol{z}_k \tag{13}$$

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

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

Example 5. $f(x) = 1 + x^2$

$f(x)$ 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. Union Bound and the Codeword Component Pattern Distance Spectrum

For a given RSC code, we have shown in the previous section that each codeword $c(x)$ is made up of $b(x)$ and $h(x)$ which have $a(x)$ as their common factor as shown in (4) and (5). Now, 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 $a(x) \in \mathcal{A}_c(d)$.

Then, the union bound of the bit-error rate can be calculated as

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} \sum_{a(x) \in \mathcal{A}_c(d)} w_H(a(x)g(x)) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{14}$$

However, since the high-weight codewords have minor contribution on the unioin bound, (14) can be further approximated by setting a limit on the maximum value of the codeword weight d_{max}, resulting in

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\text{max}}} \sum_{a(x) \in \mathcal{A}_c(d)} w_H(a(x)g(x)) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{15}$$

On the other hand, since the weight of codeword is the summation of information and parity check parts as shown in (1), when $w_H(b(x)), w_H(h(x)) \geq 2$, we have

$$\mathcal{A}_c(d) = \bigcup_{\ell=2}^{d-2} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \tag{16}$$

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 (17)

$$\mathcal{A}_c(d) \approx \mathcal{A}'_c(d) = \left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \right\} \bigcup \left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(d-\ell) \cap \mathcal{A}_h(\ell)\} \right\} \tag{17}$$

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

Example 6. $\alpha = 1$

Let $d = 7$. The $\mathcal{A}'_c(7)$ becomes

$$\begin{aligned} \mathcal{A}'_c(7) &= \Big\{ \{\mathcal{A}_b(2) \cap \mathcal{A}_h(7-2)\} \bigcup \{\mathcal{A}_b(3) \cap \mathcal{A}_h(7-3)\} \Big\} \bigcup \\ &\quad \Big\{ \{\mathcal{A}_b(7-2) \cap \mathcal{A}_h(2)\} \bigcup \{\mathcal{A}_b(7-3) \cap \mathcal{A}_h(3)\} \Big\} \\ &= \Big\{ \{\mathcal{A}_b(2) \cap \mathcal{A}_h(5)\} \bigcup \{\mathcal{A}_b(3) \cap \mathcal{A}_h(4)\} \Big\} \bigcup \\ &\quad \Big\{ \{\mathcal{A}_b(5) \cap \mathcal{A}_h(2)\} \bigcup \{\mathcal{A}_b(4) \cap \mathcal{A}_h(3)\} \Big\} \end{aligned} \tag{18}$$

If we set $d = 8$, $\mathcal{A}'_c(8)$ becomes

$$\begin{aligned}\mathcal{A}'_c(8) &= \Big\{\{\mathcal{A}_b(2) \cap \mathcal{A}_h(8-2)\} \bigcup \{\mathcal{A}_b(3) \cap \mathcal{A}_h(8-3)\}\Big\} \bigcup \\ &\quad \Big\{\{\mathcal{A}_b(8-2) \cap \mathcal{A}_h(2)\} \bigcup \{\mathcal{A}_b(8-3) \cap \mathcal{A}_h(3)\}\Big\} \\ &= \Big\{\{\mathcal{A}_b(2) \cap \mathcal{A}_h(6)\} \bigcup \{\mathcal{A}_b(3) \cap \mathcal{A}_h(5)\}\Big\} \bigcup \\ &\quad \Big\{\{\mathcal{A}_b(6) \cap \mathcal{A}_h(2)\} \bigcup \{\mathcal{A}_b(5) \cap \mathcal{A}_h(3)\}\Big\}\end{aligned} \tag{19}$$

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 $\mathcal{A}_c(8)$.

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 RSC codes. For each RSC code and a frame size of $N = 64$, the codeword is BPSK modulated and transmitted over the AWGN channel. At the receiver end, the Viterbi algorithm is used to decode before a decision is made on the decoded sequence.

The partial codeword component pattern distance spectrum for the $5/7,\ 37/21$ and $23/35$ RSC codes are shown in Tables VI, VII and VIII respectively. It is worth noting that in all the tables, the codeword components are arranged in ascending order of codeword weight, with the d_{free} components at the top of the table. From the simulation results, we observed $d_{\text{max}} = d_{\text{min}} + 3$ is a sufficient value for obtaining the BER bounds.

TABLE VI: Partial Codeword Component Pattern Distance Spectrum for the $5/7$ RSC code, $d_{\text{max}} = 8$

$a(x)$	$b(x)$	$h(x)$
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 VII: Partial Codeword Component Pattern Distance Spectrum for the $37/21$ RSC code,$d_{\text{max}} = 9$

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

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

TABLE VIII: Partial Codeword Component Pattern Distance Spectrum for the $23/35$ RSC code,$d_{\text{max}} = 10$

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

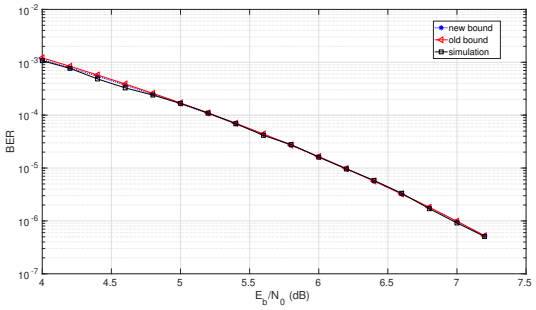

Fig. 1: Old Bound vs New Bound vs Simulation for 5/7 RSC Code

polynomial representation $1+x^2$ and can be factorized into 2 irreducible polynomials, which means 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 irreducuble polynomial and the order $\epsilon = 3$. This means 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 used in our novel method is sufficient for this 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. However, we observe from Table VII that the codeword has no parity check components of weight 3. The feedback connection has the polynomial representation $1+x^4$, which can be factorized into 4 and there are no low-codewords with systematic components of weight 3. In

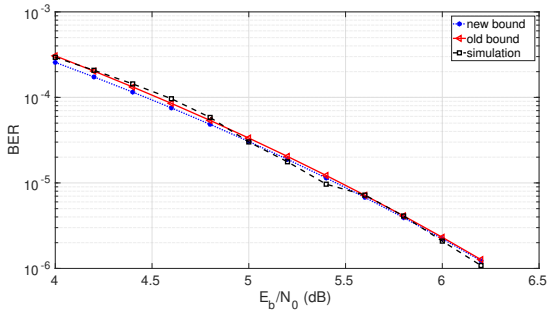

Fig. 2: Old Bound vs New Bound vs Simulation for 37/21 RSC Code

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. Again, this suggests that the approximation used in our novel method is sufficient for this RSC code.

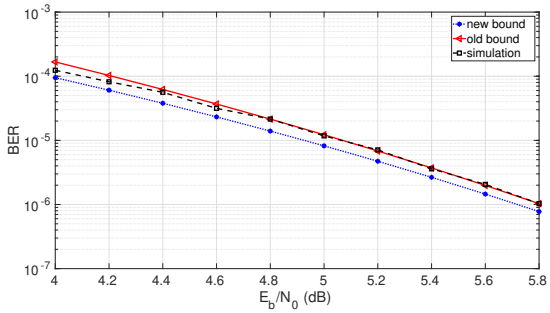

Fig. 3: Old Bound vs New Bound vs Simulation for 23/35 RSC Code

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 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 VII. 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. 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 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 listing the codeword component pattern distance spectrum for selected RSC codes up to a cut-off weight d_{max} by focusing first on codewords with systematic components such that $2 \leq w_H(b(x)) \leq 3$, and then codewords with parity check components such that $2 \leq w_H(h(x)) \leq 3$. Compared to the transfer function method, it has low complexity and provides extra information with regard to the pattern of the low-weight codeword components $b(x)$ and $h(x)$, which makes it very useful for interleaver design. We compared the bounds obtained using our novel method with the bounds 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 $w_H(h(x)), w_H(h(x)) = 4$ 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] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan and M. Jezequel, "Designing good permutations for turbo codes: towards a single model," 2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577), Paris, France, 2004, pp. 341-345, doi: 10.1109/ICC.2004.1312507.

[7] 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.