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 V, 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 VII and the paper concludes in Section VIII.

II. PRELIMINARIES

A polynomial in x, with degree M and coefficients in $\mathcal{V}$ is denoted by $v(x)$, and defined as

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

$w_H(v(x))$ denotes the Hamming weight of $v(x)$ an is equal to the number of non-zero coefficients terms.

p represents a prime number, whiles GF(p) denotes a Galois field with p. GF(p^M) is called the *extension field* of GF(p) and GF(p) is called the *ground field* of GF(p^M). $v(x)$ with its coefficients taken from GF(p), $v_M \neq 0$, is defined over GF(p) if the addition and multiplication of the coefficients are done modulo-p.

$v(x)$ is called *monic* if $v_m = 1$, *irreducible* if it is defined over GF(p) and cannot be factorised into lower degree polynomials over GF(p) and a *prime polynomial* if it is both monic and irreducible. GF(p^M) with polynomial elements is constructed by a prime polynomial $v(x)$ with degree M by considering the set of all polynomials with degree less than M, with ordinary addition and with polynomial multiplication modulo-$v(x)$, where polynomial multiplication modulo-$v(x)$ refers to dividing the product by $v(x)$ and returning the remainder.

The polynomial elements of GF(p^M) are in β, in a form such as $1 + \beta$. Alternatively, we can represent polynomial elements in power notation as $\beta^m,\ 0 \leq m \leq p^M - 1$, where β^2 may be used in place of $1 + \beta$, 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. From this point onwards, GF(p^M) represents the extension field of GF(p) with polynomial elements, unless otherwise stated.

Let β be a non-zero element of GF(2^M). Then, ϵ denotes the *order* of β, and is defined as the least positive integer value such that $\beta^\epsilon = 1$. β is a *primitive element* of GF(2^M) if $\epsilon = 2^M - 1$. A prime polynomial $v(x)$ with degree M is called a *primitive polynomial* if β is a primitive element in GF(2^M) generated by $v(x)$.

Finally, 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. Union Bound of RSC Codes

A. A Brief Review of RSC Codes

$c(x)$ denotes an RSC codeword generated using the feedforward and feedback connections of shift registers, which are determined by its generator function. For a rate $1/2$ shift register, the generator function may be written as

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

where 1 yields the systematic component of the codeword while the parity check component of the codeword is specified by $f(x)$ and $g(x)$, which represent the feedforward and feedback connections of the shift registers respectively. Then, if a input sequence $b(x)$ is fed into the shift register, $c(x)$ may be written as

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

where $b(x)$ doubles as the systematic component of the codeword and

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

denotes the parity-check component of the codeword. It is obvious that

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

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. In practice however, since higher-weight codewords have very little effect on its overall error-correcting capability, we usually use a partial distance spectrum, where the largest codeword weight value is set to d_{max}.

We consider the parity check component, and for large frame sizes, the presence of $g^{-1}(x)$ involves 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

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

Any $b(x)$ that meets the condition in (4) is called a *return-to-zero* (RTZ) input, and can be written as

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

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

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

Thus, for a given $f(x)$ and $g(x)$, we can formulate our goal as, to find all $a(x)$s which generate low-weight codeword components in (5) and (6) simultaneously.

IV. 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

$$\mathcal{A}_c(d) = \bigcup_{\ell=2}^{d-2} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \tag{7}$$

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 (18)

$$\begin{aligned} \mathcal{A}_c(d) \approx \mathcal{A}'_c(d) = &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \right\} \bigcup \\ &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(d-\ell) \cap \mathcal{A}_h(\ell)\} \right\} \end{aligned} \tag{8}$$

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

Example 1. 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

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

, 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 I: 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 II: 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	$1+x^4$	$1+x+x^2+x^3+x^4$
$1+x+x^5+x^6$	$1+x+x^4+x^6+x^9+x^{10}$	$1+x^{10}$

TABLE III: 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	$1+x^2+x^3+x^4$	$1+x+x^4$
$1+x+x^2+x^3+x^5$	$1+x^3+x^4+x^8+x^9$	$1+x^7+x^9$
$1+x+x^2+x^3+x^5+x^7+x^8$	$1+x+x^3+x^4+x^7+x^{12}$	$1+x^{11}+x^{12}$
$1+x^2+x^3+x^7+x^9+x^{10}$	$1+x^{14}$	$1+x+x^2+x^6+x^8+x^9+x^{13}+x^{14}$

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

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\text{max}}} \sum_{a(x) \in \mathcal{A}'_c(d)} w_H(a(x)g(x)) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{9}$$

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

In order to calculate $\mathcal{A}'_c(d)$, we require the knowledge of $h(x)$ and $b(x)$ when $w_H(c(x)),\ w_H(b(x)) = l$. However, since there is essentially no difference between the general structure of $h(x)$ and $b(x)$, we restrict our attention to the $h(x)$, and we present a method for determining valid values of $h(x)$ when $2 \leq w_H(h(x)) \leq 3$.

A. The Characteristic of Weight 2 Parity Check Pattern

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

$$h(x) = 1 + x^a \tag{10}$$

without any loss of generality. From (6), it is obvious that any valid $h(x)$ should have the same roots as $f(x)$. We can then reformulate our goals as to find all $h(x)$ with roots equal to those of $f(x)$. Now, we consider how to find a for 2 distinct cases of $f(x)$.

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

For this case, $f(x)$ generates GF(2^M) and the roots of $f(x)$ are taken from GF(2^M). Let β be a root of $f(x)$. Substituting β into (10), we get

$$\begin{aligned} h(\beta) &= 0 \\ 1 + \beta^a &= 0 \end{aligned} \tag{11}$$

and

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

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

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

$f(x)$ generates the field GF(2^2) where $\epsilon = 3$, 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 IV for the first four valid values of a.

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

$f(x)$ can be used used to generate GF(2^4) where $\epsilon = 5$ 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 V for the first four valid values of a.

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

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

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

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

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 of order M_k with root $\beta \in \text{GF}(2^{M_k})$, $\beta^{\epsilon_k} = 1$. For each $f_k(x)$, the valid values of a_k are such that

$$\mathcal{A}_k = \{a_k \mid a_k \bmod \epsilon_k \equiv 0\}$$

and the valid values of a are such that

$$a \in \bigcap_{k=1}^{K} \mathcal{A}_k$$

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

$f(x)$ can be written as

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

$1+x$ is prime in $GF(2)$, where $\epsilon = 1$. 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 VI for the first four valid values of a.

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

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

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

For this weight case,

$$h(x) = 1 + x^a + x^b,\ a \neq b \tag{12}$$

without loss of generality. Our goal remains the same as it was for weight 2 parity check patterns and we consider how to find the pair (a, b) for the same 2 distinct cases of $f(x)$.

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

Let β be a root of $f(x),\ \beta \in$ GF(2^M). Substituting β into (12), we get

$$\begin{aligned} h(\beta) &= 0 \\ 1 + \beta^a + \beta^b &= 0 \end{aligned} \tag{13}$$

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 $\mathcal{Z} = \{(\eta_1,\ \zeta_1), (\eta_2,\ \zeta_2), \cdots\}$. Then, any valid value $(a,\ b)$ values should satisfy the condition

$$(a, b) \equiv (\eta,\ \zeta) \bmod \epsilon,\ (\eta,\ \zeta) \in \mathcal{Z} \tag{14}$$

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

The elements of GF(2^2) are shown in Table VII 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 VII: 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 VIII: $f(x) = 1 + x + x^2$

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

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 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

$$\mathcal{AB}_k = \{(a_k,\ b_k) \mid (a_k,\ b_k) \equiv (\eta^{(k)},\ \zeta^{(k)}) \bmod \epsilon_k,\ (\eta^{(k)},\ \zeta^{(k)}) \in \mathcal{Z}_k\} \tag{15}$$

and

$$(a,\ b) \in \bigcup_{k=1}^{K} \mathcal{AB}_k \tag{16}$$

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

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

VI. 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

$$\mathcal{A}_c(d) = \bigcup_{\ell=2}^{d-2} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \tag{17}$$

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 (18)

$$\begin{aligned} \mathcal{A}_c(d) \approx \mathcal{A}'_c(d) = &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(\ell) \cap \mathcal{A}_h(d-\ell)\} \right\} \bigcup \\ &\left\{ \bigcup_{\ell=2}^{\ell+\alpha} \{\mathcal{A}_b(d-\ell) \cap \mathcal{A}_h(\ell)\} \right\} \end{aligned} \tag{18}$$

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

Example 7. 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

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

, we can list the low weight codeword component patterns for a given RSC code using (5) and (6). We set $d_{\max} = d_{\min} + 3$ 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

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\text{max}}} \sum_{a(x) \in \mathcal{A}'_c(d)} w_H(a(x)g(x)) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{19}$$

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

VII. 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 (19) 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 6, 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 irreducuble polynomial and from Example 2 and Example 5, 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 6, 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.

VIII. 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.