A Novel Method for Obtaining the Pattern of Low-Weight Codewords 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 distance spectrum of any RSC code that has the added benefit of revealing the pattern of the low-weight codewords. 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] is one of the forward-error correcting codes that comes very close to satisfying the Shannon limit for AWGN channels. It was introduced by Claude Berrou in 1993 and has been used in many applications and it has been 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 the parallel concatenation of two *recursive systematic convolutional* (RSC) codes (usually of the same kind) 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.

For this reason, interleaver design for TCs has been highly researched for many years and generally, they are grouped into random and deterministic interleavers. Random interleavers determine their order of permutation in a pseudo-random manner. TCs using random interleavers usually have good error-correcting capabilities but impose huge memory constraints for many practical applications due to the use of interleaver tables. A notable example of a random interleaver is the S-random interleaver.

On the other hand, deterministic interleavers generate their order of permutation via algorithms and as such, can be generated on the fly, and do not require permutation tables. Popular deterministic interleavers include *quadratic permutation polynomial* (QPP) interleaver [5], *almost regular permutation* (ARP) interleaver [6] and *dithered relative prime* (DRP) interleaver. A protograph based interleaver design for punctured turbo codes is also introduced in [7]. Deterministic interleavers also make it possible to perform parallel decoding once the interleaver meets certain requirements. Despite all these benefits, it is a well-known fact that in terms of TC error-correcting performance, random interleavers always outperform deterministic interleavers, especially for long frame sizes.

The design of a good deterministic interleaver requires the complete knowledge of all the low-weight codeword patterns in the component RSC codes 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 codewords. 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 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 generated by message inputs 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 codewords. 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

$v(x)$ denotes the polynomial representation of any binary sequence. Specifically, the symbols $b(x)$ and $h(x)$ represent the input message and parity-check sequence respectively in polynomial form. The Hamming weight of $b(x)$ and $h(x)$, denoted by $w_H(b(x))$ and $w_H(h(x))$ respectively, is equal to the length of the polynomial $b(x)$ or $h(x)$.

m is used to represent the order of $v(x)$ and GF(2^m) represents the extended Galois field generated by a prime polynomial which has order m. β^i represents a non-zero element in GF(2^m), $1 \leq i \leq 2^m - 1$. Any β^i s.t. $(\beta^i)^j = 1,\ j \leq 2^m - 1$ is know as a *primitive element*

Given a pair of integers $(e,\ f)$, $(e,\ f) \bmod 2^m - 1$ 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 a shift register 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 represents the systematic (input) bits of the output and $f(x)$ and $g(x)$ represent the feedforward and feedback connections of the shift register respectively.

By using the distance spectrum of an RSC code it is possible to estimate 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. Interested readers are referred to [3] where the transfer function for the $5/7$ RSC code is derived. 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 pattern distance spectrum*. Compared to the transfer function method, its complexity is independent of the number of states of the RSC. It is also able to reveal the pattern of low-weight codewords. 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, RTZ inputs and their corresponding parity-check sequences share a common factor. The second method shows how to obtain this common factor when the weight of the parity-check sequence is fixed for a given RSC code.

A. Low-weight Codewords

The parity-check sequence can be expressed as

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

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

$$b(x) \bmod g(x) \equiv 0 \tag{2}$$

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 (2) can be written as

$$b(x) = a(x)g(x) \tag{3}$$

where $a(x)$ is a monic polynomial with the coefficient of the lowest term not equal to 0. By fixing $b(x)$ from (3) into (1), we have

$$\begin{aligned} h(x) &= f(x) \cdot g^{-1}(x) \cdot a(x)g(x) \\ &= a(x)f(x) \end{aligned} \tag{4}$$

Revisiting (4), once $f(x)$ is given, our goal is to find $a(x)$ that results in a low-weight $h(x)$. To this end, we consider the roots of $f(x)$ and we can reformulate our goal as to find weight-w polynomials ($h(x)$) which take all the roots of $f(x)$ as its roots. The roots of $f(x)$ depend on its characteristic make-up and once that is known, we can easily determine the structure of $h(x)$ for

a given value of $w_H(h(x))$. The characteristic make-up of $f(x)$ can be grouped into the three cases below.

1) Single primitive polynomial.
2) Prime but not a primitive polynomial.
3) Made up of repeated polynomial roots.

We present a method for determining valid values of $h(x)$ for a given RSC code when $2 \leq w_H(h(x)) \leq 3$. It is worth noting that the method to be discussed can also be used to obtain valid values of $b(x)$, because there is no difference between the general structure of $h(x)$ and $b(x)$ once the Hamming weight is fixed.

B. Solution for $w_H(h(x)) = 2$

For this weight case, we can write $h(x)$ as $h(x) = 1 + x^a$ without any loss of generality.

a) Case1: $g(x)$ *is a single primitive polynomial:* Since $f(x)$ is a prime polynomial, it has a primitive element β as its root, which means, β is also a root of $h(x) = 1 + x^a$. Substituting β into the equation, we have

$$\begin{aligned} 1 + \beta^a &= 0 \\ \beta^a &= 1 \end{aligned} \tag{5}$$

It is obvious that a is related to the order of β and for any primitive polynomial that generates the extended field GF(2^m), its order is $2^m - 1$. This means that any valid value of a should satisfy the condition below.

$$a \bmod 2^m - 1 \equiv 0$$

Example 1. $f(x) = 1 + x + x^2$.
$f(x)$ generates the field GF(2^2) where the order of β is 3. 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.

b) Case2: $f(x)$ is prime polynomial but not primitive

Similar to the case for primitive polynomials, we need to find the order of β. For fields generated by prime polynomials, there is a value $j < 2^m - 1$ such that

$$\beta^j = 1, \ j \mid 2^m - 1$$

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

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

where j is the order. Therefore, any valid value of a should satisfy the condition below.

$$a \bmod j \equiv 0$$

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

$f(x)$ can be used used to generate GF(2^4) and in the field it generates, the order of β is 5. 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: $f(x) = 1 + x + x^2 + x^3 + x^4$

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

c) Case3: $f(x)$ is made up of repeated polynomial roots.

We may write $f(x)$ as

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

The kth polynomial is prime and its order is $j_k,\ j_k \leq 2^m - 1$. Because each value of j_k is unique, valid values of a should satisfy the condition below.

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

For the special case where $f(x)$ has equal repeated roots, the above condition simplifies to

$$a \bmod jK \equiv 0$$

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 in this field 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: $f(x) = 1 + x^2$

$a(x)$	$b(x)$
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. Solution for $w_H(h(x)) = 3$

For this weight case, $f(x) = 1 + x^a + x^b,\ a \neq b$ without loss of generality.

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

Because $f(x)$ is a prime polynomial, it has a primitive element β as its root, which means that β is also a root of $q(x) = 1 + x^u + x^v$. Substituting β into the equation, we have

$$\begin{aligned} 1 + \beta^a + \beta^b &= 0 \\ \beta^a + \beta^b &= 1 \end{aligned} \tag{6}$$

By refering to the table of the extended field for GF(2^m), we can find the valid (e, f) pairs s.t. $\beta^e + \beta^f = 1,\ e \neq f$. If there are no values for the pair (e, f), then there is no $h(x)$ such that $w_H(h(x)) = 3$ for the given $f(x)$. We represent the set of (e, f) pairs as $\boldsymbol{z} = \{(e_1, f_1), (e_2, f_2), \cdots\}$. Then any valid value $(a,\ b)$ values should satisfy the condition

$$(a, b) \equiv (e, f) \bmod 2^m - 1,\ (e, f) \in \boldsymbol{z}$$

since $\beta^{2^m - 1} = 1$.

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

The elements of GF(2^2) are shown in Table IV and it is obvious that there is exactly 1 valid

(e, f) pair s.t. $\beta^e + \beta^f = 1$ 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: $r(x)$ is prime but not a primitive polynomial

Similar to the case where $f(x)$ is primitive, we confirm the existence of (e, f) pairs s.t. $\beta^e + \beta^f = 1$. If there are no values for the pair (e, f), then there is no $h(x)$ such that $w_H(h(x)) = 3$ for the given $f(x)$. We represent the set of (e, f) pairs as $\boldsymbol{z} = \{(e_1, f_1), e_2, f_2), \cdots\}$. Then any valid value for a and b should satisfy the condition

$$(a, b) \equiv (e, f) \bmod j, \ (e, f) \in \boldsymbol{z}$$

where j is the order of β.

Example 5. $f(x) = 1 + x + x^2 + x^3 + x^4$

From the table of the extended field generated by $f(x)$ (Table VI), we see that there are no valid $(e, \ f)$ and as such $h(x)$ s.t. $w_H(h(x)) = 3$ is non-existent for $f(x) = 1 + x + x^2 + x^3 + x^4$.

c) Case3: $f(x)$ is made up of repeated polynomial roots

We may write $g(x)$ as

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

TABLE VI: Non-zero Elements of GF(2^4) generated by $f(x) = 1 + x + x^2 + x^3 + x^4$

power	polynomial
$\beta^0 = \beta^5 = \beta^{10} = \beta^{15} = 1$	1
$\beta = \beta^6 = \beta^{11}$	β
$\beta^2 = \beta^7 = \beta^{12}$	β^2
$\beta^3 = \beta^8 = \beta^{13}$	β^3
$\beta^4 = \beta^9 = \beta^{14}$	$\beta^3 + \beta^2 + \beta + 1$

For the kth polynomial, we refer to the (extended) field table and determine if there exists any $(e^{(k)},\ f^{(k)})$ pairs s.t. $\beta^{e^{(k)}} + \beta^{f^{(k)}} = 1$. If it exists for all K polynomials, then the valid values for u and v should satisfy the condition

$$\bigcap_{k=1}^{K}(u, v) = (ku', kv'),\ (u, v) \equiv (e^{(k)}, f^{(k)}) \bmod 2^m - 1,\ (e^{(k)}, f^{(k)}) \in \boldsymbol{z}^{(k)}$$

A special case is when $f(x)$ has equal repeated roots. The above condition simplifies to

$$(a, b) = (Ka', Kb'),\ (a, b) \equiv (e, f) \bmod 2^m - 1,\ (e, f) \in \boldsymbol{z}$$

where $(a',\ b')$ is the valid pair value for the single polynomial root.

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

V. Union Bound and the Codeword Pattern Distance Spectrum

Having determined how to find valid values of $b(x)$ and $h(x)$ for Hamming weights ≤ 3, we are now in a position to generate the codeword pattern distance spectrum for a given RSC code. We take a union bound like approach towards the generation of the codeword pattern distance spectrum. The approach is outlined below.

1) Beginning with (3), we find all values of $b(x),\ w_H(b(x)) = 2$ that have the same roots as $g(x)$ and divide $g(x)$ by each valid polynomial to obtain the corresponding $a(x)$.
2) Then using (4), we multiply each $a(x)$ by $f(x)$ to obtain the corresponding value of $h(x)$. It is worth noting that $w_H(h(x))$ may be $\geq w_H(b(x))$.

3) Since we are interested in only the low weight codewords, we ignore any $b(x)$ s.t. $w_H(b(x))+ w_H(h(x)) \geq d_{\text{max}}$.
4) Next we set the weight value of $b(x)$ to $w_H(b(x)) = 3$, and repeat steps 1 and 2 while ignoring $b(x)$ that meet the condition in step 3.
5) To obtain a complete codeword pattern distance spectrum, we do a reverse operation, *i.e.* we focus on (3) and find all values of $g(x),\ w_H(\boldsymbol{h}) = 2$ that have the same roots as $f(x)$ and divide $f(x)$ by each valid polynomial to obtain the corresponding $a(x)$.
6) Then using (3), we repeat steps 2 through 4, being careful to avoid repitition.
7) Finally we arrange all valid values of $b(x)$ and $h(x)$ in ascending value of codeword weight,$w_H(b(x)) + w_H(h(x))$.

The codeword pattern distance spectrum for the $5/7,\ 37/21$ and $23/35$ RSC codes are shown in Tables VII, VIII and IX respectively.

TABLE VII: Partial Structured 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 VIII: Partial Structured 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	$1+x^4$	$1+x+x^2+x^3+x^4$
$1+x+x^5+x^6$	$1+x+x^4+x^6+x^9+x^{10}$	$1+x^{10}$

We use the codeword pattern distance spectrum to calculate the bit-error bounds for each RSC and compare them to the bit-error bounds obtained via the distance spectrum as well as

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

simulation results. We use the probability of bit-error in doing this and a more general formula for calculating P_b is shown below [4]:

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} w(d) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{7}$$

where $w(d) = \sum_{i=1}^{\infty} i\ a(d,i)$ and $a(d,i)$ is the number of codewords of weight d generated by an input message of weight i. If we set a limit on the maximum value of the codeword weight d_{max} we can rewrite (7) as

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\text{max}}} w(d) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{8}$$

From the simulation results, we observed $d_{\text{max}} = d_{\text{min}} + 3$ is a sufficient value for obtaining the BER bounds.

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.

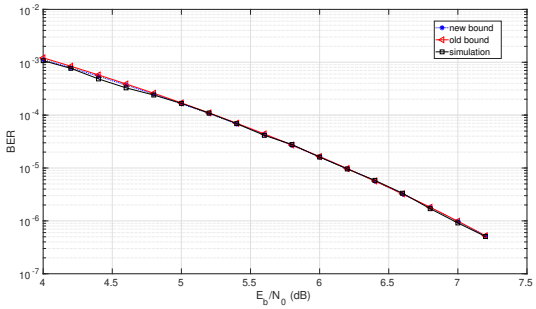

Fig. 1: Old Bound vs New Bound vs Simulation for 5/7 RSC Code

We observe that for both Fig. 1 and Fig. 2, 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 codewords generated considering $b(x),\ w_H(b(x)) > 3$ as well as codewords which have a parity-check sequence $h(x),\ w_H(h(x)) > 3$ do not have much effect on the BER of the code as E_b/N_0 increases.

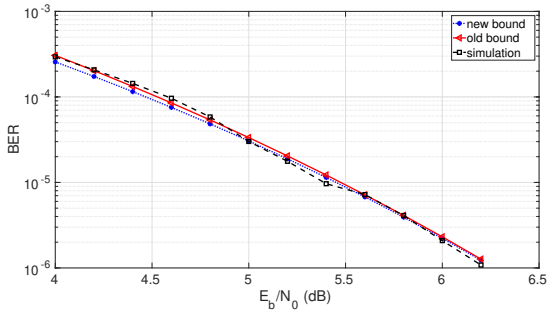

Fig. 2: Old Bound vs New Bound vs Simulation for 37/21 RSC Code

However, in Fig. 3, it is apparent that codewords generated by higher weight RTZs as well as those with parity-check sequences with weights $w_H(b(x)) > 3$ still dominate at higher E_b/N_0 values and need to be considered. As can be observed from the graph, there is a very distinct gap between the new bound and the old bound. Moreover, the bounds do not converge as E_b/N_0 increases. However, the old bounds and simulation results converge as the E_b/N_0 value increases.

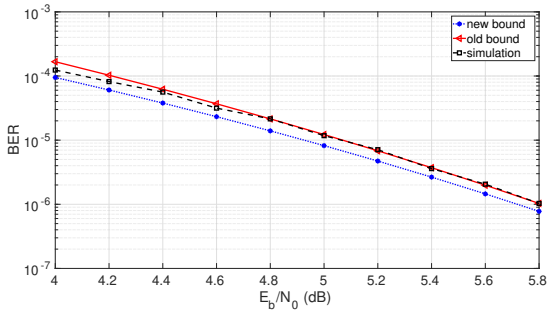

Fig. 3: Old Bound vs New Bound vs Simulation for 23/35 RSC Code

VII. Conclusion

In this paper, we presented a method for listing the codeword pattern distance spectrum for selected RSC codes up to a cut-off weight d_{max} by focusing on codewords generated by the RTZ inputs of weight $w_H(b(x)) \leq 3$ as well as codewords with parity-check sequences $w_H(h(x)) \leq 3$. Compared to the transfer function method, it has low complexity and provides extra information with regard to the structure of the RTZ inputs as well as the parity-check sequences, 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. The results suggest that considering codewords generated by RTZ inputs of weight $w_H(b(x)) > 3$ as well as codewords with parity-check sequences $w_H(h(x)) > 3$ will improve the accuracy of the bounds obtained via our method.

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.