A Novel Method for Obtaining the Structure of RTZ Inputs and their Corresponding Parity-Check Sequences: A First Step Towards Interleaver Design

Kwame Ackah Bohulu

January 8, 2021

0.1 Abstract

The knowledege of the distance spectrum, as well as the structure of the message inputs that make up the distance spectrum for a specific Recursive Systematic Convolutional (RSC) code is vital to the design of Turbo Code interleavers. Whiles the distance spectrum of a RSC code can be obtained by calculationg its transfer function, it does not provide any information about the structure of the message inputs.

In this paper,we present a novel low-complexity method for determining the distance spectrum of any RSC code which has the added benefit of revealing the structure of the Return-To-Zero (RTZ) inputs that make up the distance spectrum as well as their corresponding parity-check sequences. We then go a step further and present a method for deriving a general polynomial representation for both RTZ inputs and parity-check sequences with a Hamming weight of up to 4 for any RSC code.

Combining these two methods, we list the partial distance spectrum for selected RSC codes up to a cut-off weight $d_{\max}$ and compare simulation results to the bounds obtained via our novel method and the regular method.

0.2 Introduction

The Turbo Code (TC) [**?**], is known for being one of the forward-error correcting codes that come very close to satisfying the Shannon limit for AWGN channels. It was introduced by Claude Berrou in 1993 and steadily gained fame and has been used in many applications including Mobile and Satelite communication networks. The simplest and most common construction of the TC is via the parallel concatenation of two Recursive Systematic Convolutional (RSC) codes (of the same kind) via an interleaver. The reason why the TC has such a good error correcting capability has been attributed to the low multiplicity of its minimum distance codeword and through many years of intensive research in the field, it is common knowledge that this is largely due to the use of the interleaver in the TC construction.

Interleaver design for TCs has been a hot topic 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 and therefore require interleaver tables in both the transmitter and the receiver. Even thought TCs made with random interleavers have really good error-correcting capabilities (especially for medium and long frame sizes), the need for interleaver tables imposes huge memory constraints for many practical applications. A notable example of a random interleaver is the S-random interleaver.

On the flip side, deterministic interleavers generate their order of permutation via algorithims and as such can be generated on the fly, killing the need for permutation tables. Popular deterministic interleavers include Quadratic Permutation Polynomial (QPP) interleaver [**?**], Almost Regular Permutation (ARP) Interleaver and the Dithered Relative Prime (DRP) interleaver. Deterministic interleavers also make it possible to perform parallel decoding, once the interleaver meets certain requirements. Given all these benefits, it is a well known fact that in terms on TC error-correcting performace random interleavers always outperform deterministic interleavers, especially for long frame sizes. Another benefit of using deterministic interleavers is the ability to custom design the interleaver to a specific component code to improve the overall error-correcting capability of the TC.

The most common approach to deterministic interleaver design is the minimum free distance (d_{free}) maximization approach, where the interleaver is designed with the aim to maximize the value of d_{free}. This approach whilst simplistic, has produced some good interleavers only after considering higher weight inputs [**?**]. Therfore, using it as a general rule of thumb for all deterministic interleaver design approaches might not be the best, especially when the minimum distance codeword for the component code is generated by an input message with weight greater than 2.

To better design custom interleavers for a specific component code requires deep knowledge of its distance spectrum, more specifically the general structure of the message inputs that make up the distance spectrum. These message inputs are such that they diverge from and then return to the initial stateand are refered to as Return-To-Zero (RTZ) inputs There are many methods available for obtaining the distance spectrum of an RSC code including finding the transfer function of the RSC code [**?**]. However, the transfer function method is not a very helpful tool when it comes to custom interleaver design since aside information about the number of codewords of weight d generated by a message input of weight w, it provides zero information about the structure of the RTZ inputs. As an added downside, the complexity of calculating the transfer function for a given RSC code increases with the number of states.

In this paper,we present a novel alternate method to the Transfer Function whose complexity is independent of the number of states of the component code, and has the added benefit of making known the structure of the RTZ inputs that make up the distance spectrum. With the

knowledge of the structure of the message inputs, we derive a general polynomial representation for them based on the weight of the message input after which we go a step further and derive corresponding parity-weight equations for the codewords they generate. Finally, a summary of the structure of the message inputs as well as their corresponding parity weight equations is tabulated for different RSC codes.

0.3 Preliminaries

Binary vectors are represented by bold font and $\dot{\boldsymbol{v}}$ represent an infinite repitition of the vector $\boldsymbol{v}$ whiles $(\boldsymbol{v})_j$ represents the repitition of vector $\boldsymbol{v}$ j times

$\boldsymbol{\phi}$ represent a primitive element in the extended Galois field GF(2^τ) and the vector representation for all elements in GF(2^τ) are written as $\boldsymbol{\phi}_i,\ 0 \leq i \leq 2^\tau$.

The subscript i represents the decimal value of the binary vector, which means $\boldsymbol{\phi}_0$ represents the all zero vector. All addition and multiplication operations are done in GF(2^τ).

0.4 Recursive Systematic Convolutional Codes:Review

An (n, k) RSC code is a convolutional code generated by using feedback shift registers which has its input bits as part of the codeword. At each time instant it receives an input of k bits and outputs n bits. The output bits are determined by the generator function, which may be written in polynomial notation as $\left[1 \ \frac{f(x)}{g(x)}\right]$. Given the systematic nature of the RSC code, we only focus on the parity part of the RSC code and write the generator function simply as $\left[\frac{f(x)}{g(x)}\right]$ where $f(x)$ and $g(x)$ represent the feedfoward and feedback connections of the RSC encoder.

The impulse response caused by the feedback connection of the RSC encoder is easily calculated as

$$\phi_g(x) = g^{-1}(x)$$

Its equivalent vector representation is

$$\dot{\boldsymbol{\phi}}_g$$

$\boldsymbol{\phi}_g$ is known as the cycle of the RSC code and τ is the cycle length. The impulse response of the RSC code is calculated as

$$\phi(x) = f(x)\phi_g(x)$$

By careful observation, we can choose two integer values h and r such that we can write the equivalent vector representation as

$$\boldsymbol{\phi} = \boldsymbol{\phi}_h \dot{\boldsymbol{\phi}}_r$$

The minimum distance ($d_{\min}$) of the RSC code determines its error-correcting capability. With the aid of the distance spectrum, it is possible to determine $d_{\min}$ as well as its multiplicity. The most common way to find the distance spectrum is via the transfer function of the RSC code. The transfer function enumerates all the paths that diverge from and then return to the initial state [**?**], ie the RTZ inputs. In other words, the distance spectrum provides information about the number of codewords of weight d generated by an RTZ input of weight w. Combined with the knowledge that all RTZ inputs are divisible by $G(D)$ [**?**], we introduce a novel method for determining what we refer to as the *input structure distance spectrum* of any RSC code in the next section. This version of the distance spectrum is a better tool for interleaver design compared to the regular distance spectrum.

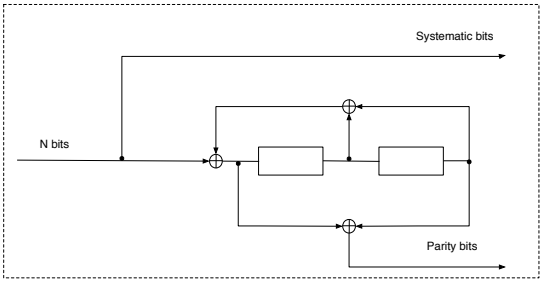

Figure 0-1: $[\frac{1+x^2}{1+x+x^2}]$ RSC Encoder

Example 1. An RSC encoder is shown in Figure 0-1 with $k = 1$ and $n = 2$. Its generator function is given by $[\frac{1+x^2}{1+x+x^2}]$ which may be written as 5/7 in octal form where 5 and 7 correspond to the numerator and denominator of the generator function respectively. For this RSC code, The cycle $\boldsymbol{\phi}_g = \boldsymbol{\phi}_6$ with a cycle length of $\tau = 3$. Also, $\boldsymbol{\phi} = (1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ \cdots)$ which may be written in terms of the elements of GF(8) as $\boldsymbol{\phi}_7\ \dot{\boldsymbol{\phi}}_3$.

Moving foward all other examples and discussions relating to RSC codes will be done using the 5/7 RSC code unless otherwise stated.

0.5 Novel Method For Finding The Distance Spectrum

In this section, we present our novel method for obtaining what we have named the *structured distance spectrum*. Our novel method can be seen to be the combination of two different but related methods. The first method is quite simple and makes use 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 RTZ input or its parity-check sequence is fixed for a given RSC code. After explaining the inner working of our novel method, we use it to obtain the partial structured distance spectrum for the 5/7, 37/21 and 23/35 RSC codes.

Throughout this section, the symbols $\boldsymbol{b}$, $\boldsymbol{h}$ and $\boldsymbol{c}$ represent the input message, parity-check sequence and the codeword respectively in vector form whiles $b(x)$, $h(x)$ and $c(x)$ are their equivalent polynomial representation. We use the polynomial r(x) when there is no need to distinguish between $b(x)$ and $h(x)$. Also when we need to refer to $g(x)$ and $h(x)$ in general terms we use the polynomial $q(x)$. $\boldsymbol{r}$ (and) $\boldsymbol{q}$ refer to their corresponding vector representations.

0.5.1 Low-weight Codewords

In polynomial notation, the parity-check sequence can be expressed as

$$h(x) = f(x) \cdot g^{-1}(x) \cdot b(x) \tag{0-1}$$

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

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

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 (0-2) can be written as

$$b(x) = a(x)g(x) \tag{0-3}$$

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

$$\begin{aligned} h(x) &= f(x) \cdot g^{-1}(x) \cdot a(x)g(x) \\ &= a(x)f(x) \end{aligned} \tag{0-4}$$

From (0-3) and (0-4), we observe that $a(x)$ is a common factor in both equations and if we are able to solve for $a(x)$ via either of the equations, the remaining equation can be solved. To solve for $a(x)$ requires that in either equations it should be the only unknown variable. At first glance it might seem like $g(x)$ and $h(x)$ are the only known variables since they are dependent on the RSC code in question. However, is we remember that the weight of $h(x)$ and $b(x)$ is directly proportional to the number of terms it has, then we are on our way to obtaining our second known variable. All that stands in our way is determining valid power values for the polynomial terms, depending on the weight of $h(x)$ or $b(x)$.

Revisiting (0-4), we see that $h(x)$ is divisible by both $f(x)$ and $a(x)$ and therefore their roots are also the roots of $h(x)$. However, since the value of $a(x)$ is variable, we will use the root of $f(x)$ when determining the valid power values for the polynomial terms of $h(x)$ for a given weight

value. Similarly, $b(x)$ is also divisible by $a(x)$ and $g(x)$ (from (0-3)) but we utilize the roots of $g(x)$ when determining the valid power values for the polynomial terms of $b(x)$ for a given weight value. Having determined the values of $h(x)$ or $b(x)$ for a fixed Hamming weight value and RSC code, we can now generate the partial structured distance spectrum for that RSC code. Beginning with (0-3), we solve for $a(x)$ for a predetermined number of valid values of $b(x)$ and then use $a(x)$ in (0-4) to obtain the corresponding value of $h(x)$. Since there is no guarantee that low-weight message inputs always map to low-weight parity-check sequences, we repeat the same operation obtaining $a(x)$ via (0-4) for a predetermined number of valid values of $h(x)$ and then use $a(x)$ in (0-3) to obtain the corresponding value of $b(x)$. From the lists obtained from both operations, we can then generate the partial structured distance spectrum by selecting $h(x)$ and $b(x)$ values s.t. $w_H(\boldsymbol{c}) \leq d_{\max},\ d_{\max} = d_{\text{free}} + 3$.

0.5.2 Finding Valid Values of $h(x)$ and $b(x)$ for a Fixed Hamming Weight and RSC Code

We have established that the relationship between the roots of $f(x)$ and $h(x)$ as well as the relationship between the roots of $g(x)$ and $b(x)$. Depending on the characteristic makeup of $f(x)$, we can easily determine the structure of $h(x)$ for the its different weight values. The same logic applies to determining the structure of $b(x)$ for its different weight values from $g(x)$. We therefore present a method for determining valid values of $h(x)$ and $b(x)$ for a given RSC code when the weight values $w_H(\boldsymbol{h}),\ w_H(\boldsymbol{b}) \leq 3$. We state here that there is no RTZ input of weight 1, since it goes against the definition of RTZ inputs.Consequently, there also cannot be a corresponding parity-check sequence of weight 1 and we can ignore those cases. The characteristic makeup of both $f(x)$ and $g(x)$ can be grouped into the 4 cases below.

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

Since there is no difference between the general structure of $h(x)$ and $b(x)$ once the Hamming weight is fixed, we use $q(x)$ to represent either of them. If we fix $r(x)$ into (0-4) or (0-4), then $f(x)$ will be equivalent to$g(x)$ so we will use $r(x)$ to represent either of them.

A. Structure of $q(x),\ w_H(\boldsymbol{q}) = 2$

It is obvious that the structure of $q(x) = 1 + x^a$ when $w_H(\boldsymbol{h}) = 2$. What remains is finding the right value for a.

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

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

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

For any primitive polynomial that generates the extended field GF(2^m), $\beta^{2^m-1} = 1$ which means any valid of a should satisfy the condition below.

$$2^m - 1 \mid a$$

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

Similar to the case for primitive polynomials, there is a value $j < 2^m - 1$ such that

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

. Therefore any valid value of a should satisfy the condition below.

$$j \mid a$$

Case3: $r(x)$ is made up of equal repeated polynomial roots.

Given the above condition we have, $r(x) = (r_p^{o_p}(x))^k$, where $r_p^{o_p}(x)$ represents the prime polynomial $r(x)$ can be factorized into and k is the number of times it is repeated. $r_p^{o_p}(x)$ has β as its roots and we need to find d s.t. $\beta^j = 1$ in the (extended) field it generates. Since $\beta^{kj} = 1$ in that same (extended) field , the valid values for a should satisfy the condition below

$$kj \mid a$$

Case4: $r(x)$ is made up of unique repeated polynomial roots.

We may write $r(x)$ as

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

The kth polynomial is prime and has a value d_k s.t. $\beta^{j_k} = 1$. Since each value of j_k is unique, valid values of a should satisfy the condition below.

$$\bigcap_{k=1}^{K} \{j_k \mid a\}$$

Example 2. 5/7 RSC Code $f(x) = 1 + x^2,\ g(x) = 1 + x + x^2$
$f(x)$ is a match for Case3, ie it is made up of equal repeated polynomial roots. It can be written as

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

$1 + x$ is prime in $GF(2)$ and $\beta^1 = 1$. Since $k = 2$, we have $\beta^k = \beta^2 = 1$. $h(x) = 1 + x^a$ and the valid values of $a = \{2, 4, 6, \cdots\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the table below for the first 4 valid values of a

Table 1: 5/7 RSC Code, $f(x) = 1 + x^2$

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

$g(x)$ is a match for Case1, ie it is a primitive polynomial for GF(2^2). In this extended field, $\beta^3 = 1$. $b(x) = 1 + x^b$ and the valid values of b are $b = \{3, 6, 9, \cdots\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the table below for the first 4 valid values of b.

Table 2: 5/7 RSC Code, $g(x) = 1 + x + x^2$

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

Example 3. 37/21 RSC Code, $f(x) = 1 + x + x^2 + x^3 + x^4,\ g(x) = 1 + x^4$
$f(x)$ is a match for Case2, ie it is a prime polynomial but not primitive. It can be used to generate GF(2^4) and in the field it generates, $\beta^5 = 1$. Therefore, the valid values of a are $a = \{5, 10, 15, \cdots\}$. The corresponding values for $h(x)$ and $a(x)$ are shown in the table below for the first 4 valid values of a.

Table 3: 37/21 RSC Code, $f(x) = 1 + x + x^2 + x^3 + x^4$

$a(x)$	$b(x)$	$h(x)$
$1 + x$	$1 + x + x^4 + x^5$	$1 + x^5$
$1 + x + x^5 + x^6$	$1 + x + x^4 + x^6 + x^9 + x^{10}$	$1 + x^{10}$
$1 + x + x^5 + x^6 + x^{10} + x^{11}$	$1 + x + x^4 + x^6 + x^9 + x^{11} + x^{14} + x^{15}$	$1 + x^{15}$
$1 + x + x^5 + x^6 + x^{10} + x^{11} + x^{15} + x^{16}$	$1 + x + x^4 + x^6 + x^9 + x^{11} + x^{14} + x^{16} + x^{19} + x^{20}$	$1 + x^{20}$

$g(x)$ is a match for Case3, ie it is made up of equal repeated polynomial roots. It can be written as

$$g(x) = (1 + x)^4,\ k = 4$$

$1 + x$ is prime in $GF(2)$ and $\beta^1 = 1$. Since $k = 4$, we have $\beta^k = \beta^4 = 1$. $b(x) = 1 + x^b$ and the valid values of $b = \{4, 8, 12, \cdots\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the table below for the first 4 valid values of a

Table 4: 37/21 RSC Code, $g(x) = 1 + x^4$

$a(x)$	$h(x)$	
1	$1 + x^4$	$1 + x + x^2 + x^3 + x^4$
$1 + x^4$	$1 + x^8$	$1 + x + x^2 + x^3 + x^5 + x^6 + x^7 + x^8$
$1 + x^4 + x^8$	$1 + x^{12}$	$1 + x + x^2 + x^3 + x^5 + x^6 + x^7 + x^9 + x^{10} + x^{11} + x^{12}$
$1 + x^4 + x^8 + x^{12}$	$1 + x^{16}$	$1 + x + x^2 + x^3 + x^5 + x^6 + x^7 + x^9 + x^{10} + x^{11} + x^{13} + x^{14} + x^{15} + x^{16}$

Example 4. 23/35 RSC Code, $f(x) = 1 + x + x^4,\ g(x) = 1 + x^2 + x^3 + x^4$
$f(x)$ is a match for Case1, ie it is a primitive polynomial which can be used to generate the extended field GF(2^4). In this field, $\beta^{15} = 1$. $h(x) = 1 + x^a$ and the valid values of a are $a = \{15, 30, 45, \cdots\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the table below for the first 2 valid values of a.

Table 5: 23/35 RSC Code, $f(x) = 1 + x + x^4$

$a(x)$	$b(x)$	$h(x)$
$1 + x^2 + x^3 + x^5 + x^7 + x^8 + x^{11}$	$1+x+x^3+x^4+x^7+x^{11}+x^{12}+x^{13}+x^{14}+x^{15}$	$1 + x^{15}$
$1 + x^2 + x^3 + x^5 + x^7 + x^8 + x^{11} + x^{15} + x^{16} + x^{17} + x^{18} + x^{20} + x^{22} + x^{23} + x^{26}$	$1 + x + x^3 + x^4 + x^7 + x^{11} + x^{12} + x^{13} + x^{14} + x^{16} + x^{18} + x^{19} + x^{22} + x^{26} + x^{27} + x^{28} + x^{29} + x^{30}$	$1 + x^{30}$

$g(x)$ is a match for Case4, ie it is made up of unique repeated polynomial roots. It be written as

$$g(x) = (1 + x)(1 + x + x^3),\ K = 2$$

$1 + x$ is prime in $GF(2^1)$ and $\beta^1 = 1$. $1 + x + x^3$ is prime in $GF(2^3)$ and $\beta^7 = 1$. $b(x) = 1 + x^b$ and consequently, the valid values of b that meet the condition

$$\bigcap_{k=1}^{K} \{j_k \mid b\}$$

are $b = \{7, 14, 21, \cdots\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the table below for the first 4 valid values of b

Table 6: 23/35 RSC Code, $g(x) = 1 + x^2 + x^3 + x^4$

$a(x)$	$b(x)$	$h(x)$
$1 + x^2 + x^3$	$1 + x^7$	$1 + x + x^2 + x^6 + x^7$
$1 + x^2 + x^3 + x^7 + x^9 + x^{10}$	$1 + x^{14}$	$1 + x + x^2 + x^6 + x^8 + x^9 + x^{13} + x^{14}$
$1 + x^2 + x^3 + x^7 + x^9 + x^{10} + x^{14} + x^{16} + x^{17}$	$1 + x^{21}$	$1+x+x^2+x^6+x^8+x^9+x^{13}+x^{15}+x^{16}+x^{20}+x^{21}$
$1 + x^2 + x^3 + x^7 + x^9 + x^{10} + x^{14} + x^{16} + x^{17} + x^{21} + x^{23} + x^{24}$	$1 + x^{28}$	$1 + x + x^2 + x^6 + x^8 + x^9 + x^{13} + x^{15} + x^{16} + x^{20} + x^{22} + x^{23} + x^{27} + x^{28}$

B. Structure of $q(x),\ w_H(\boldsymbol{q}) = 3$

Again, it is obvious that $q(x) = 1 + x^u + x^v, = v \neq u$ and given $r(x)$, our task is to find valid values for u and v. If there are no values for the pair (u, v), then $q(x)$ s.t. $w_H(\boldsymbol{q}) = 3$ does not exist for the given $r(x)$.

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

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

$$\begin{aligned} 1 + \beta^u + \beta^v &= 0 \\ \beta^u + \beta^v &= 1 \end{aligned} \tag{0-6}$$

where $u \neq v$ For any extended field GF(2^m), $m > 1$, there are exactly $q = 2^{(m-1)} - 1$ (e, f) pairs s.t. $\beta^e + \beta^f = 1,\ e \neq f$. This can easily be confirmed from the table of elements representing GF(2^m). If we represent the set of (e, f) pairs as $\boldsymbol{z} = \{(e_1, f_1), e_2, f_2), \cdots, (e_1, f_q)\}$, then any valid value for u and v should satisfy the condition

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

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

Case2: $r(x)$ is prime but not a primitive polynomial

A prime but not primitive polynomial generates an (extended) field with j elements, where $j < 2^m - 1$. Similar to the case where $r(x)$ is primitive, we confirm the existence of (e, f) pairs s.t. $\beta^e + \beta^f = 1$. If we represent the set of (e, f) pairs as $\boldsymbol{z} = \{(e_1, f_1), e_2, f_2), \cdots, (e_1, f_q)\}$, then any valid value for u and v should satisfy the condition

$$(u, v) \equiv (e, f) \bmod j,\ (e, f) \in \boldsymbol{z}$$

since $\beta^j = 1$

Case3: $r(x)$ is made up of equal repeated polynomial roots

For this case, we can write $r(x)$ as $r(x) = (r_p^{o_p}(x))^k$, where $r_p^{o_p}(x)$ represents the prime polynomial that is the root of $r(x)$ and k is the number of times it is repeated. If $r_p^{o_p}(x)$ is a primitive polynomial and $m > 1$, then from Case1 there are exactly $q = 2^{(m-1)} - 1$ $(e,\ f)$ pairs s.t. $\beta^e + \beta^f = 1,\ e \neq f$ in the set $\boldsymbol{z}$. If $r_p^{o_p}(x)$ is prime but not a primitive polynomial, then we determine the number of elements in the set $\boldsymbol{z}$ directly from the table representing the field it generates. If we focused on $f_p^{o_p}(x)$ only, (u', v') should satisfy the condition

$$(u', v') \equiv (e, f) \bmod 2^m - 1,\ (e, f) \in \boldsymbol{z}$$

. However, since $f(x)$ is made up of $f_p^{o_p}(x)$ repeated k times, and $\beta^{ke} + \beta^{kf} = 1$, the valid values for u and v should satisfy the condition

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

Case4 $r(x)$ is made up of unique repeated polynomial roots

We may write $r(x)$ as

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

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

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

Example 5. 5/7 RSC Code, $f(x) = 1 + x^2,\ g(x) = 1 + x + x^2$

$f(x)$ is a match for Case3 and can be written as $(1 + x)^2$. $(1 + x)$ is a primitive polynomial for GF(2). The elements of GF(2) are shown in Table **??** and in this field, there are no valid (e, f). Therefore, $h(x)$ s.t. $w_H(\boldsymbol{h}) = 3$ does not exist.

$g(x)$ is a match for Case1, ie it is a primitive polynomial for GF(2^2) with $\beta^3 = 1$. The elements of GF(2^2) are shown in Table 5 and it is obvious that there is exactly 1 ($q = 2^{(2-1)} - 1 \ = 1$) valid (e, f) pair s.t. $\beta^e + \beta^f = 1$ and that is $(1, 2)$. Which means the valid values of (u, v) pairs are any values s.t. $(u, v) \equiv (1, 2) \bmod 3$. The corresponding values for $a(x)$, $b(x)$ and $h(x)$ are shown in the table below for the first 4 valid values of (u, v)

Table 7: Non-zero Elements of GF(2^2) generated by $g(x) = 1 + x + x^2$

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

Table 8: 5/7 RSC, $g(x) = 1 + x + x^2$

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

Example 6. 37/21 RSC Code, $f(x) = 1 + x + x^2 + x^3 + x^4,\ g(x) = 1 + x^4$
$f(x)$ is a match for Case2, ie it is a prime but not primitive polynomial. From the table of the extended field generated by $f(x)$ (Table 9), we see that there are no valid $(e,\ f)$ and as such $h(x)$ s.t. $w_H(\boldsymbol{h}) = 3$ is non-existent.

Table 9: Non-zero Elements of GF(2^2) 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
$\beta^4 = \beta^9 = \beta^{14}$	$\beta^3 + \beta^2 + \beta + 1$

$g(x)$ is a match for Case3, ie it is made up of equal repeated roots. After factorization, we have $g(x) = (1 + x)^4$.(1+x) is a primitive polynomial which generates the field GF(2) and in this field, there are also no valid $(e,\ f)$ and as such $b(x)$ s.t. $w_H(\boldsymbol{b}) = 3$ is also non-existent.

Example 7. 23/35 RSC Code, $f(x) = 1 + x + x^4,\ g(x) = 1 + x^2 + x^3 + x^4$
$f(x)$ is a match for Case1, ie it is a primitive polynomial which can be used to generate the extended field GF(2^4) with $\beta^{15} = 1$.. The elements of GF(2^4) are shown in Table 10 and we can see that there are 7 valid (e, f) pairs

Which means the valid values of (u, v) pairs are any values s.t. $(u, v) \equiv (e, f) \bmod 15, (e, f) \in \boldsymbol{z},\ \boldsymbol{z} = \{(1, 4),\ (2, 8),\ (3, 14),\ (5, 10),\ (6, 13),\ (7, 9),\ (11, 12)\}$. The corresponding values for $a(x),\ b(x)$ and $h(x)$ are shown in the Table 11 below for the first 4 valid values of (u, v)

Table 10: Non-zero Elements of GF(2^4) generated by $f(x) = 1 + x + x^4$

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

$g(x)$ is a match for Case4, ie it is made up of unique repeated roots. Upon factorizing we have $g(x) = (1 + x)(1 + x + x^3)$. Table 12 shows the elements of GF(2^3) generated by $(1 + x + x^3)$ and we can confirm that there are 3 valid $(e,\ f)$ pairs. However since there are no valid $(e,\ f)$

Table 11: 23/35 RSC, $f(x) = 1 + x + x^4$

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

pairs in GF(2), it also means that there cannot be any valid $(u,\ v)$ pairs and $b(x)$ s.t. $w_H(\boldsymbol{b}) = 3$ does not exist.

Table 12: Non-zero Elements of GF(2^3) generated by $1 + x + x^3$

power	polynomial
$\beta^0 = \beta^7 = 1$	1
β	β
β^2	β^2
β^3	$\beta + 1$
β^4	$\beta^2 + \beta$
β^5	$\beta^2 + \beta + 1$
β^6	$\beta^2 + 1$

0.5.3 Partial Structured Distance Spectrum

The partial structured distance spectrum for the 5/7, 37/21 and 23/35 RSC codes are shown in Tables 13, 14 and 15 respectively.

Table 13: Partial Structured Distance Spectrum for the 5/7 RSC code,$d_{\max} = 8$

$a(x)$	$b(x)$	$h(x)$
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 14: Partial Structured Distance Spectrum for the 37/21 RSC code,$d_{\max} = 9$

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

Table 15: Partial Structured Distance Spectrum for the 23/35 RSC code,$d_{\max} = 10$

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

We use the partial structured distance spectrum to calculate bit-error bounds for each RSC and compare them to the bit-error bounds obtained via the distance spectrum as well as simulation results. We use probability of bit-error in doing this and a more general formula for calculating P_b is shown below [4]

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} w(d) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{0-7}$$

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 (0-7) as

$$P_b \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{d_{\max}} w(d) Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{0-8}$$

0.6 Simulation Results

Figures 0-2, 0-3 and **??** show the comparison of the bounds obtained via our novel method to bounds obtained using the transfer function method as well as simulation results. For each RSC code, 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.

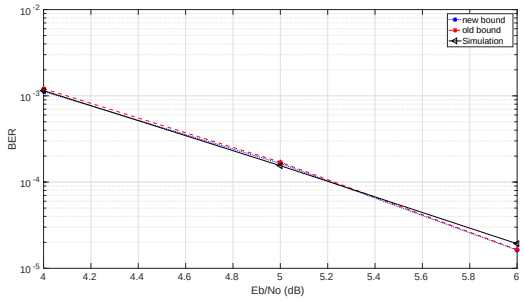

Figure 0-2: Old Bound vs New Bound vs Simulation for 5/7 RSC Code

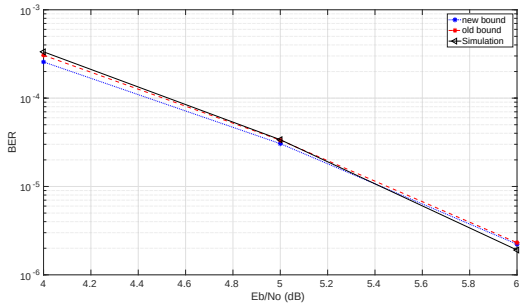

Figure 0-3: Old Bound vs New Bound vs Simulation for 37/21 RSC Code

0.7 Conclusion

In this paper, we presented a method for listing input message which produce codewords with low-weight parity bit sequences for a for a given (n, k) RSCC. Compared to the Transfer function method, it has low complexity and provides more information about distance spectrum of the RSCC. Using a specially configured finite state machine, we can obtain a partial distance spectrum which we use to calculate an upper bound for the RSCC.

Bibliography

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes", Proc. Intern. Conf. Communications (ICC), Geneva, Switzerland, pp. 1064- 1070, May 1993.

[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",
Chapter 3, John Wiley & Son.

[5] Jing Sun, Oscar Y. Takeshita "Interleavers for Turbo Codes Using Permutation Polynomials over Integer Rings", IEEE Trans. Inform. Theory, vol. 51, pp. 101 - 119 Jan. 2005.

[6] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel "Designing Good Permutations for Turbo Codes: Towards a Single Model",IEEE Communications Society 2004,pp.341-345