A Novel Low-Complexity Method for Revealing The Structure of The Message Inputs of the Distance Spectrum For Recursive Systematic Convolutional Codes

Kwame Ackah Bohulu

November 18, 2020

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 provides no information about the structure of the message inputs and the complexity involved in calculating the transfer function increases with the number of states of the RSC code.

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 message 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, we compare the upper bound for both methods to simulation results and it is revealed that the upper bound obtained by the novel method is much tighter.

0.2 Introduction

A Convolutional Code (CC) is generated by passing an input message through a linear finite-state shift register. The structure of this code is such that it is best described using a trellis. This structure makes it possible to employ soft decision decoding algorithms, the most popular of these algorithms being the Viterbi algorithm. CC are used extensively in mobile communication and space communication application as a major component in concatenated code. Depending on the configuration of the shift register being used to generate the code, a CC can either be *recursive* or *nonrecursive*. In the case of the recursive CC, a feedback shift register is used to generate the code. Furthermore if the input message appears in the CC, it is known as *systematic*. Recursive Sytematic Convolutional (RSC) codes are used as component codes for turbo codes, which are one of the few error correcting codes with performance very close to the Shannon limit [1].

Low weight codeword are produced when the parity bit sequence has a very low weight. Amongst all such codewords, the one with the lowest weight determines the free distance d_{free} of the code. d_{free} of a RSC code is a very important factor and determines its error-correction performance [4]. This can be obtained from the distance spectrum of the RSC code which requires the calculation of the transfer function. The distance spectrum provides information about the number of codewords of weight d generated by a message input of weight w. The message inputs are such that they diverge from and then return to the initial state, assuming edge effect is ignored. These message inputs are referred to as Return-To-Zero (RTZ) inputs and with respect to interleaver design for turbo codes, the structure of these inputs makes it possible to design good interleavers. For this reason, the distance spectrum obtained as a result of the transfer function is not very useful, since it provides no 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, we compare the upper bound for both methods to simulation results and it is revealed that the upper bound obtained by the novel method is much tighter.

The rest of the research paper is organised as follows. In Section 0.3, we give a brief review of RSC codes. In section 0.4, we describe how the distance spectrum of a RSC Code is obtained using the transfer function, followed by the presentation of our novel method in Section 0.5. Simulation results are presented in Section **??** and we draw conclusions in Section 0.7

0.3 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 D notation as $\left[1 \ \frac{F(D)}{B(D)}\right]$. 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 $\left[\frac{F(D)}{B(D)}\right]$ where $F(D)$ and $B(D)$ represent the feedfoward and feedback connections of the RSC encoder.

It should be noted that the prescence of the feedback loop means that the code produced will have an infinite-length impulse response denoted $\boldsymbol{t}$. A parameter which arises as a result of this infinite impulse response is know as the cycle length[5], denoted by τ, which is defined as the length of the output cycle $\boldsymbol{p}$ of an RSC encoder when the input is $(1\ 0\ 0\ 0\ 0\ \cdots)$.The impulse response $\boldsymbol{t}$ and by extension the cycle $\boldsymbol{p}$ are unique for each RSC code.

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

A RSC encoder is shown in Figure 0-1. Its generator function is given by $[\frac{1+D^2}{1+D+D^2}]$ which may be written as 5/7 in octal form where 5 and 7 correspond to the numerator and denomenator of the generator function respectively. For the 5/7 RSC code, $\boldsymbol{t} = [1110110110...]$ which gives us a cycle $\boldsymbol{p}$ of (1 1 0) and a cycle length $\tau = 3$. Also $k = 1$ and $n = 2$. Moving foward all examples and discussions relating to RSC codes will be done using the 5/7 RSC code unless otherwise stated.

0.4 Distance Spectrum via Transfer Function of RSC Code

The distance spectrum of a code gives information about codeword weights and the number of such codewords present in the code. For the RSC code, this 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. Based on the method described in [3], we outline the process involved in deriving the transfer function of the 5/7 RSC code.

Figure 0-2: State Diagram of the 5/7 RSC code

First, the state diagram of the 5/7 RSC code is redrawn as shown in Figure 0-2. The zero state is split into two and the transition from the zero state to itself is ommited. On each edge, the variables Y^d and X^w are used to represent the output weight d and input weight w of the path respectively. We treat each edge label as a transfer block and we employ the following rules for graph simplification. Assume two edge labels H and B

1. If H is connected in series to B, the labels are merged as HB
2. If H is connected in parallel to B, the labels are merged as $H + B$
3. If the edges are in a feedback configuration, where H and B are the feedfoward and feedback portions respectively, the labels are merged as $\frac{H}{1-HB}$

In the following, we demonstrate the process for deriving the transfer function for the RSCC shown in Figure 0-2. The respective state diagram transformations are also shown in Figure 0-3.

Figure 0-3: State Diagram Transformations involved in Transfer Function Calculation

Example 1. Define all edge labels
edge $a \triangleq 00 \rightarrow 10$, edge $b \triangleq 10 \rightarrow 01$
edge $c \triangleq 01 \rightarrow 00$, edge $d \triangleq 01 \rightarrow 10$
edge $e \triangleq 10 \rightarrow 11$, edge $f \triangleq 11 \rightarrow 11$
edge $g \triangleq 11 \rightarrow 10$

1. Simplify edge g with feedback configuration

$$\frac{1}{1-YX}$$

2. Merge edges e, f, g with series configuration and rename it edge h

$$\text{edge } h = Y \times \frac{1}{1-YX} \times Y = \frac{Y^2}{1-YX}$$

3. Merge edges h, b with parallel configuration and rename it edge j

$$\text{edge } j = YX + \frac{Y^2}{1-YX} = \frac{YX + Y^2X^2 + Y^2}{1-YX}$$

4. Merge edges j, d with feedback configuration and rename it edge k

$$\begin{aligned}\text{edge } k &= \frac{\frac{YX+Y^2X^2+Y^2}{1-YX}}{1-(\frac{YX+Y^2X^2+Y^2}{1-YX})} \\ &= \frac{Y(X-YX^2+Y)}{1-2YX+Y^2X^2-Y^2}\end{aligned}$$

5. Calculate transfer function by merging edges k, a, c with series configuration

$$\begin{aligned}T(Y,X) &= Y^2X \times \frac{Y(X-YX^2+Y)}{1-2YX+Y^2X^2-Y^2} \times Y^2X \\ &= \frac{Y^5X^2(Y-YX^2+Y)}{1-2YX+Y^2X^2-Y^2} \\ &= Y^5X^3 + Y^6(X^4+X^2) + Y^7(X^5+3X^3) + \\ &Y^8(X^6+6X^4+X^2) + Y^9(X^7+10X^5+5X^3) + ...\end{aligned}$$

From the example, it is clear that the complexity involved in deriving the transfer function increases as the number of states of the RSCC increases. Also to obtain the distance spectrum requires an extra division operation. Furthermore, with this method, it is impossible to get any extra detail regarding the structure of the input message bits which correspond to a codeword of a particular weight. In the next section, we present a novel method whose complexity is independent of the number of states of the RSC code. As an added bonus, information regarding the structure of the RTZ inputs can be obtained using this method.

0.5 Novel Method for Finding the Distance Spectrum

As mentioned earlier, our interest is to determine if a given input message **b** is an RTZ input. In this section, we present a novel low-complexity method for determining just that and all that is required is the knowledge of the feedback connection of the RSC code $G(D)$. According to [**?**], every RTZ input is a multiple of $g(x)$. Also, the transfer function is used to enumerate that paths that diverge from and return to the initial state of the trellis, ie the path taken by RTZ inputs. With the combination of the above knowledege, it is possible to develop a method to not only determine which RTZ inputs produce low-weight parity sequences, but the structure of such RTZ inputs as well. For convinience sake we will switch to polynomial representation, where $g(x)$ and $f(x)$ represent the feedback and feedforward connections of the RSC code respectively. Also $b(x)$ and $h(x)$ are the polynomial representations of the input message and its corresponding parity-bit sequence respectively.

Simply put, $b(x)$ is an RTZ input if

$$b(x) \bmod g(x) = 0 \tag{0-1}$$

For an RTZ that meets that criteria $h(x)$ is accurately calculated as

$$h(x) = \frac{b(x)}{g(x)} f(x) \tag{0-2}$$

Our method for finding the distance spectrum for a given RSC code is outlined below

1. Initialize an array $\boldsymbol{b}(x)$ with the initial elements set to $1 + x$ and 1. Also initialize 2 empty arrays $\boldsymbol{r}(x)$, $\boldsymbol{h}(x)$ which will be used to store the RTZ inputs and their corresponding $h(x)$ respectively.

2. For each element in $\boldsymbol{b}(x)$ if the condition in 0-1 is met, save the polynomial in $\boldsymbol{r}(x)$. If there is a need, find $h(x)$ using 0-2 and save it in $\boldsymbol{h}(x)$

3. All elements that meet the condition in 0-1 are then deleted from $\boldsymbol{b}(x)$ and the remaining elements are duplicated which doubles the size of the array. The polynomial length is either extended or maintained by adding an extra "0" or "1" to the binary representation of the polynomial.

4. Repeat steps 2 and 3 untill the binary representation of the elements in $\boldsymbol{r}(x)$ has length M

To reduce the time required to use this method, we an extra criteria where in step 3 we either delete elements in $\boldsymbol{b}(x)$ with a weigth ¿w or codeword weight of d or both. For $M = 16$ we find all $b(x)$ and $h(x)$ for which $w_H(\mathbf{h}) = 2$ and $w_H(\mathbf{h}) = 4$. The results are shown in Table 1 and 2 respectively . For $w_H(\mathbf{h}) = 4$, only the first 10 results are shown. Also 3 shows all $b(x)$ and $h(x)$ that produce a codeword weight $w_H \leq 8$ for $M = 32$.

Table 1: codewords with parity bit sequence weight $w_H(\mathbf{h}) = 2$

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

Table 2: codewords with parity bit sequence weight $w_H(\mathbf{h}) = 4$

$w_H(\mathbf{b})$	$b(x)$	$h(x)$
2	$1 + x^3$	$1 + x + x^2 + x^4$
3	$1 + x^2 + x^4$	$1 + x + x^3 + x^4$
	$1 + x^4 + x^5$	$1 + x + x^2 + x^5$
	$1 + x + x^5$	$1 + x^3 + x^4 + x^5$
4	$1 + x^2 + x^3 + x^5$	$1 + x + x^4 + x^5$
	$1 + x^2 + x^5 + x^6$	$1 + x + x^3 + x^6$
	$1 + x^4 + x^6 + x^7$	$1 + x + x^2 + x^7$
	$1 + x + x^4 + x^6$	$1 + x^3 + x^5 + x^6$
	$1 + x + x^6 + x^7$	$1 + x^3 + x^4 + x^7$
	$1 + x + x^3 + x^7$	$1 + x^5 + x^6 + x^7$

Table 3: All $b(x)$ which produce codewords with weight $w_H \leq 8$ for $M = 32$

$w_H(\mathbf{b})$	$b(x)$	$h(x)$
5	$1 + x + x^2$	$1 + x^2$
6	$1 + x^3$	$1 + x + x^2 + x^3$
	$1 + x + x^3 + x^4$	$1 + x^4$
7	$1 + x^2 + x^4$	$1 + x + x^3 + x^4$
	$1 + x + x^5$	$1 + x^3 + x^4 + x^5$
	$1 + x^4 + x^5$	$1 + x + x^2 + x^5$
	$1 + x + x^3 + x - 5 + x^6$	$1 + x^6$
8	$1 + x^2 + x^3 + x^5$	$1 + x + x^4 + x^5$
	$1 + x^6$	$1 + x + x^2 + x^4 + x^5 + x^6$
	$1 + x + x^4 + x^6$	$1 + x^3 + x^5 + x^6$
	$1 + x^2 + x^5 + x^6$	$1 + x + x^3 + x^6$
	$1 + x + x^3 + x^7$	$1 + x^5 + x^6 + x^7$
	$1 + x + x^6 + x^7$	$1 + x^3 + x^4 + x^7$
	$1 + x^4 + x^6 + x^7$	$1 + x + x^2 + x^7$
	$1 + x + x^3 + x^5 + x^7 + x^8$	$1 + x^8$

0.6 Generalization of RTZ inputs and Their Corresponding Parity-Weight Equations

With our method described in the previous section, we are able to not only determine the distance spectrum of the RSC code, but we have information with regards to the structure of the RTZ inputs. In this section, provide general information with regards to the type of RTZ inputs present in a RSC code. Then, we go a step further and provide a general representation (in polynomial form) of the RTZ inputs grouped by their weight. Finally we derive equations for calculating the parity weight for these RTZ inputs. The generalization as well as the derived equations will be very useful when it comes to designing interleavers for turbo codes.

0.6.1 Preliminaries

To aid in our proofs the following have been defined with respect to the impulse response of the 5/7 RSC code. Let

$$\boldsymbol{\phi}_1 = (0\ 0\ 1) \quad , \ \boldsymbol{\phi}'_1 = (0\ 1\ 0), \ \boldsymbol{\phi}''_1 = (1\ 0\ 0) \tag{0-3}$$

$$\boldsymbol{\phi}_2 = (0\ 1\ 1) \quad , \ \boldsymbol{\phi}'_2 = (1\ 1\ 0), \ \boldsymbol{\phi}''_2 = (1\ 0\ 1) \tag{0-4}$$

$$\boldsymbol{\phi}_3 = (1\ 1\ 1) \tag{0-5}$$

To simplify calculation, we have included an addition table for all the vectors which is shown in Table 4

	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}_3$
$\boldsymbol{\phi}_1$	$\mathbf{0}_3$	–	–	–	–	–	–
$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}_2$	$\mathbf{0}_3$	–	–	–	–	–
$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}'_2$	$\mathbf{0}_3$	–	–	–	–
$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}_3$	$\mathbf{0}_3$	–	–	–
$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}_3$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}''_2$	$\mathbf{0}_3$	–	–
$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_3$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}_2$	$\mathbf{0}_3$	–
$\boldsymbol{\phi}_3$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_1$	$\mathbf{0}_3$

Table 4: Truth Table

0.6.2 Types of RTZ inputs

Regardless of the component code used in turbo coding, the RTZ inputs can be grouped into two basic forms. We shall refer to them as *base RTZ inputs* and *compound RTZ inputs*. The number of base RTZ inputs depends on the component code and cannot be broken down into 2 or more RTZ inputs. Compound RTZ inputs as the name implies, are formed from 2 or more base RTZ inputs and therefore can be broken down into base RTZ input form.

For the 5/7 component code, its base RTZ inputs are weight-2 RTZ inputs (W2RTZs) and weight-3 RTZ inputs (W3RTZs). Every RTZ input with a weight higher than 3 is a compound RTZ input. In general for RTZs with weight w greater than 3, if $w \bmod 2 = 0$, then the RTZ is made up of $w/2$ W2RTZs. On the other hand, if $w \bmod 2 = 1$, then the RTZ is made up of $\lfloor w/2 \rfloor - 1$ W2RTZs and 1 W3RTZ.

0.6.3 General Form of RTZ Inputs

In literature for turbo codes, it has been shown as the weight of the inputs increases, it has less effect on the bound of the BER[reference needed]. For this reason, for any RSC, we will provide a generalization for at most the first 4 RTZ inputs in the distance spectrum. These generalizations are obtained from using our novel method to list all the RTZ inputs of weight w that generate a codeword with weight $w_H \leq d$ and taking note of the pattern that exists. The examples shown here are done with respect to the 5/7 RSC and the at the end of the section shows the generalization for a few other RSC codes.

General Form of W2RTZs

A W2RTZ has the general form

$$\begin{aligned} P(x) &= x^{h\tau+t}(1+x^{\alpha\tau}) \\ &= x^t(x^{h\tau}+x^{(h+\alpha)\tau}) \end{aligned} \tag{0-6}$$

where

$$h = 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha = 1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - h,\ t = (0,1,...,\tau-1)$$

Proof. The proof will be to show that any polynomial of this form is an RTZ input. This can be done either by dividing $P(x)$ by $g(x)$ or by using the impulse response

division by g(x) Without loss of generality, we can assume that $h = t = 0$ and $P(x) = 1+x^{\alpha\tau}$. For any value of α, $P(x) \bmod g(x) = 0$ is always true. This proves that $P(x)$ is a RTZ input which has weight 2.

Using the impulse response If we write $P(x)$ in binary for we have a summation of vectors that take the form

$$\begin{matrix} (\mathbf{0}_3\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}'_2)_{\alpha-1}\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2) \\ (\mathbf{0}_3\ \ (\mathbf{0}_3)_\alpha\ \ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2) \end{matrix}$$

Regardless of the value of α we realize that there is an infinite summation of $\boldsymbol{\phi}_2 + \boldsymbol{\phi}_2 = \mathbf{0}_3$ after the second 1 bit leading to a low-weight codeword, proving again that that $P(x)$ is a RTZ input which has weight 2 or a W2RTZ.

This ends the proof. □

General Form of W3RTZs

A W3RTZ has the general form

$$\begin{aligned} Q(x) &= x^{h\tau+t}(1+x^{\beta\tau+1}+x^{\gamma\tau+2}) \\ &= x^{h\tau+t}+x^{(h+\beta)\tau+t+1}+x^{(h+\gamma)\tau+t+2}. \end{aligned} \tag{0-7}$$

where

$$\begin{aligned} h &= 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \beta = 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1 \\ \gamma &= 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ t = (0,1,...,\tau-1) \end{aligned}$$

Notice that $h \leq \beta$ or $h \leq \gamma$ is not a necessary condition.

Proof. Similarly we prove that $Q(x)$ is a RTZ input either dividing $Q(x)$ by $g(x)$ or utilizing the impulse response.

division by g(x) Without loss of generality, we can assume that $h = t = 0$ and $Q(x) = 1 + x^{\beta\tau+1} + x^{\gamma\tau+2}$. For any value of β and γ, $Q(x) \bmod g(x) = 0$ is always true. This proves that $Q(x)$ is a RTZ input which has weight 3.

Using the impulse response If we write $Q(x)$ in binary for we have a summation of vectors that take the form

$$\begin{array}{lll} (\mathbf{0}_{3(\gamma+h)} & \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2' \ \cdots) \\ (\mathbf{0}_{3(\beta+h)} & \boldsymbol{\phi}_2 & \boldsymbol{\phi}_2'' \ \cdots) \\ (\mathbf{0}_{3h} & \boldsymbol{\phi}_3 & \boldsymbol{\phi}_2 \ \cdots) \end{array}$$

Regardless of the value of β and γ we realize that there is an infinite summation of $\boldsymbol{\phi}_2 + \boldsymbol{\phi}_2 = \mathbf{0}_3$ after the third 1 bit leading to a low-weight codeword, proving again that that $Q(x)$ is a RTZ input which has weight 3 ie a W3RTZ.

This ends the proof.

□

.

General Form of W4RTZs

A W4RTZ has the general form

$$\begin{aligned} R(x) &= x^{h\tau+t}(1 + x^{\alpha_1\tau}) + x^{h'\tau+t'}(1 + x^{\alpha_2\tau}) \\ &= x^t(x^{h\tau} + x^{(h+\alpha_1)\tau}) + x^{t'}(x^{h'\tau} + x^{(h'+\alpha_2)\tau}) \end{aligned} \tag{0-8}$$

where

$$h,\ h' = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h$$
$$\alpha' = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h',\ t,\ t' \in \{0, 1, ..\tau - 1\}, h\tau + t \neq h'\tau + t'$$

It is obvious that the W4RTZ representation is valid since it is a combination of 2 W2RTZs

W5RTZs : Definitions and Breaking them

A W5RTZ has the general form

$$\begin{aligned} S(x) &= x^{h\tau+t}(1 + x^{\alpha\tau}) + x^{h'\tau+t'}(1 + x^{\beta\tau+1} + x^{\gamma\tau+2}) \\ &= x^t(x^{h\tau} + x^{(h+\alpha)\tau}) + x^{h'\tau+t'} + x^{(h'+\beta)\tau+t'+1} + x^{(h'+\gamma)\tau+t'+2} \end{aligned} \tag{0-9}$$

where

$$h,\ h' = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha' = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h'$$
$$\beta = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \gamma = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1$$
$$t,\ t' \in \{0, 1, ..\tau - 1\}, h\tau + t \neq h'\tau + t'$$

Again, it is obvious that the W4RTZ representation is valid since it is a combination of a W2RTZ and a W3RTZ.

0.6.4 Parity Weight Equations for RTZ Inputs

Once the general form of an RTZ input is know, we can calculate the parity weight of the codeword generated. In this section, we derive the equations for calculating the parity weight for the parity-bit sequences generated by those RTZ Inputs.

Parity Weight Equations for W2RTZs

The parity weight for a W2RTZ $w_p^{(2)}$ is given by

$$w_p^{(2)} = 2\alpha + 2 \tag{0-10}$$

Proof. For W2RTZ we consider the summation of the vectors below

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\cdots\ \boldsymbol{\phi}_2') \\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2') \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3'\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3) \end{array}$$

The derived vector will be

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_3)$$

The parity weight $w_p^{(2)}$ is given by

$$\begin{aligned} w_p^{(2)} &= 2(\alpha-1)+4 \\ &= 2\alpha+2 \end{aligned}$$

This ends the proof. □

Hamming Weight for W3RTZ Turbo Codewords

The parity weight for a W3RTZ $w_p^{(3)}$ is given by

$$2l + 2 \tag{0-11}$$

where $l = \max(\beta, \gamma)$

Proof.
The polynomial representation of a weight-3 RTZ input is given by

$$Q(x) = x^{h\tau+t}(1 + x^{\beta\tau+1} + x^{\gamma\tau+2})$$

With reference to the impulse response of the 5/7 RSC encoder,

Now, we consider the weight of the vector derived by the sumation of the followings vectors.

$$\begin{array}{ll} (\mathbf{0}_{3(\gamma+h)} & \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\mathbf{0}_{3(\beta+h)} & \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ (\mathbf{0}_{3h} & \boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \cdots) \end{array}$$

Without loss of generality, we can assume that all weight-3 RTZ inputs begin at the 0th position, ie $h = t = 0$. This is because the case where $h > 0$ or $t > 0$ is just a right-shifted version of the weight-3 RTZ. With this assumption, we we only need to consider cases where $h = 0,\ \gamma \geq h$.

Furthermore, we consider 4 general cases for all possible values of i, j, k where $i \geq k$ These cases are $(=\ \ =)$, $(=\ \ <)$, $(<\ \ =)$ and $(<\ \ <)$

Case 0: $\gamma = \beta = h$

For this case, the vectors to sum will be

$$\begin{array}{r} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ (\boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_2''\ \mathbf{0}_3\ \cdots) \end{array}$$

and the derived vector will be $(\boldsymbol{\phi}_2''\ \mathbf{0}_3\ \cdots)$ with a weight of $w_p = 2$

Case 1a: $\gamma = h < \beta$

vectors to sum:

$$\begin{array}{r} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots) \\ (\mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ +(\boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_2'\ (\boldsymbol{\phi}_2'')_{\beta-h-1}\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots)$
Parity weight:

$$w_p = 2(\beta - h) + 2 = 2\beta + 2 \tag{0-12}$$

Case 1b: $\beta = h < \gamma$

vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3\ \cdots\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2''\boldsymbol{\phi}_2''\ \cdots) \\ +(\boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_1''\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_1''\ (\boldsymbol{\phi}_2)_{\gamma-h-1}\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots)$
Parity weight:

$$w_p = 2(\gamma - h) + 2 = 2\gamma + 2 \tag{0-13}$$

Case 2a: $h < \gamma = \beta$
vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3 \cdots \ \cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \ \cdots) \\ (\mathbf{0}_3 \cdots \ \cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \ \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots) \\ \hline (\boldsymbol{\phi}_3 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1 \ \mathbf{0}_3 \ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_3 \ (\boldsymbol{\phi}_2)_{\gamma-h-1} \ \boldsymbol{\phi}_1 \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$w_p = 2(\gamma - h) + 2 = 2\gamma + 2 \tag{0-14}$$

Case 3a: $h < \gamma < \beta$
vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3 \cdots\cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \ \cdots \ \phi_2' \ \phi_2' \ \phi_2' \cdots) \\ (\mathbf{0}_3 \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \cdots) \\ \hline (\boldsymbol{\phi}_3 \ \boldsymbol{\phi}_2 \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2'' \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_3 \ (\boldsymbol{\phi}_2)_{\gamma-h-1} \ \boldsymbol{\phi}_1' \ (\boldsymbol{\phi}_2'')_{\beta-\gamma-1} \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$\begin{aligned} w_p &= 2(\gamma - h) + 2 + 2(\beta - i) \\ &= 2(\beta - h) + 2 \\ &= 2\beta + 2 \end{aligned} \tag{0-15}$$

Case 3b: $h < \beta < \gamma$

$$\begin{array}{r} (\mathbf{0}_3 \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \cdots) \\ (\mathbf{0}_3 \ \cdots\cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \ \cdots \ \phi_2'' \ \phi_2'' \ \phi_2'' \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \cdots) \\ \hline (\boldsymbol{\phi}_3 \ \boldsymbol{\phi}_2 \cdots \ \boldsymbol{\phi}_2 \ \mathbf{0}_3 \ \boldsymbol{\phi}_2' \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_3 \ \mathbf{0}_3 \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_3 \ (\boldsymbol{\phi}_2)_{j-k-1} \ \mathbf{0}_3 \ (\boldsymbol{\phi}_2')_{i-j-1} \ \boldsymbol{\phi}_3 \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$\begin{aligned} w_p &= 2(\beta - h) + 1 + 2(\gamma - \beta) + 1 \\ &= 2(\gamma - h) + 2 \\ &= 2\gamma + 2 \end{aligned} \tag{0-16}$$

From all the above cases we can conclude that the parity weight for a weight-3 RTZ sequence may be calculated as

$$w_p^{(3)} = 2l + 2 \tag{0-17}$$

where $l = \max(\gamma, \beta)$
This ends the proof $\square$

Hamming weight for W4RTZ Turbo Codewords

The parity weight for a W4RTZ $w_p^{(4)}$ is given by

$$w_p^{(4)} = \begin{cases} 2(\alpha+\alpha')+4, & \text{if } h\tau+t < (h+\alpha)\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' \\ 2(\alpha), & \text{if } h\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' < (h+\alpha)\tau+t,\ t \neq t' \\ 2(\alpha'+(h'-h)+(t'-t)-1), & \text{if } h\tau+t < h'\tau+t' < (h+\alpha)\tau+t < (h'+\alpha')\tau+t',\ t \neq t' \end{cases} \tag{0-18}$$

Proof. For all the proofs, we will rely on the parity vector for W2RTZs, which is given by

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_3)$$

Case 1: $h\tau+t < (h+\alpha)\tau+t < h'\tau+t' < (h'+\alpha')\tau+t'$

For this case, we have the following parity vector summation.

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3) \\ +(\mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \cdots\ \mathbf{0}_3) \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \cdots\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \cdots\ \mathbf{0}_3) \end{array}$$

The derived parity vector is then $(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_3\ \cdots\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha'-1}\ \boldsymbol{\phi}_3)$ and the weight for the derived parity vector is calculated as

$$\begin{aligned} w_p =& 2(\alpha)+2+2(\alpha')+2 \\ =& 2(\alpha+\alpha')+4 \end{aligned} \tag{0-19}$$

Case 2: $h\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' < (h+\alpha)\tau+t,\ t' \neq t$

For the above case, there are 3 possible vector summations as shown below

Case 2a

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3) \\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1''\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3) \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_1'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3) \end{array} \tag{0-20}$$

for Case 2a, the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2'')_{(\alpha'-1)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{((h-h')+(\alpha-\alpha')-2)}\ \boldsymbol{\phi}_3)$$

with a corresponding weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(\alpha'-1)+1+2((h-h')+(\alpha-\alpha')-2)+4 \\ &= 2(h'-h)+1+2\alpha'-1+2(h-h')+2(\alpha-\alpha') \\ &= 2(\alpha'-\alpha'+\alpha) \\ &= 2\alpha \end{aligned}$$

Case 2b

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3)\\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \mathbf{0}_3\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\boldsymbol{\phi}_3)\end{array} \tag{0-21}$$

For Case 2b, the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_2''\ (\boldsymbol{\phi}_2)_{(\alpha-1)}\ \mathbf{0}_3\ (\boldsymbol{\phi}_2')_{((h-h')+(\alpha-\alpha')-2)}\ \boldsymbol{\phi}_3)$$

And the parity weight is

$$\begin{aligned} w_p &= 2(h'-h)+1+2(\alpha'-1)+2+2((h-h')+(\alpha-\alpha')-2)+3\\ &= 2(h'-h)+1+2\alpha'-2+2+2(h-h')+2(\alpha-\alpha')-1\\ &= 2(\alpha'-\alpha'+\alpha)\\ &= 2\alpha \end{aligned}$$

Case 2c

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3)\\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \mathbf{0}_3)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3)\end{array} \tag{0-22}$$

For Case 2c, $t = t'$ and the derived vector as well as the parity weight is the same as that of the Type1 W4RTZ. Therefore for Case 2, the parity weight is given by

$$w_p = 2\alpha \tag{0-23}$$

Case 3: $h\tau + t < h'\tau + t' < (h+\alpha)\tau + t < (h'+\alpha')\tau + t',\ t' \neq t$

Similarly, there are 3 possible vector summations as shown below

Case 3a

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_3\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1'')\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_1''\ \boldsymbol{\phi}_2\cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1'')\end{array} \tag{0-24}$$

For Case 3a the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2'')_{(h-h'+\alpha)-2}\ \boldsymbol{\phi}_1''\ (\boldsymbol{\phi}_2)_{((h'-h)+(\alpha'-\alpha))}\ \boldsymbol{\phi}_1'')$$

with a weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(h-h'+\alpha-2)+2+2((h'-h)+(\alpha'-\alpha))+1\\ &= 2(h'-h)+2(\alpha-\alpha+\alpha')+1-1\\ &= 2((h'-h)+\alpha') \end{aligned}$$

Case 3b

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ (\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\end{array} \tag{0-25}$$

For Case 3b the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}'_2)_{(h'-h)}\ \boldsymbol{\phi}''_2\ (\boldsymbol{\phi}_2)_{(h-h'+\alpha)-2}\ \boldsymbol{\phi}'_1\ (\boldsymbol{\phi}''_2)_{((h'-h)+(\alpha'-\alpha))}\ \boldsymbol{\phi}'_2)$$

with a weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(h-h'+\alpha-2)+3+2((h'-h)+(\alpha'-\alpha))+2\\ &= 2(h'-h)+2(\alpha-\alpha+\alpha')+2\\ &= 2((h'-h)+\alpha'+1)\end{aligned}$$

Case 3c

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ (\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_3)\\ \hline \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_3)\end{array} \tag{0-26}$$

For Case 3c, $t = t'$ and the derived vector as well as the parity weight is the same as that of the Type1 W4RTZ.

The parity weight equations for cases 3a and 3b are different but without loss of generality if we assume that $t' > t,\ t = 0$ we see that case 2a corresponds to the case where $t' - t = 1$ whiles case 2b corresponds to the case where$t' - t = 2$ Therefore for Case 3 the parity weight is given by

$$w_p^{(4)} = 2(\alpha' + (h'-h) + (t'-t) - 1) \tag{0-27}$$

This ends the proof □

Hamming weight for W5RTZ Turbo Codewords

Proof. For each W5RTZs case, we confirm the equations for all possible W3RTZ cases,given that the W2RTZ structure is constant.

Case1: $h'\tau + t' < (h'+\beta')\tau + t' < (h'+\gamma')\tau + t' < h\tau + t < (h+\alpha)\tau + t$

Since the W2RTZ and the W3RTZ do not overlap, it is obvious that the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2(l) + 2 + 2(\alpha) + 2\\ &= 2(l+\alpha') + 4\end{aligned}$$

Case2: $h'\tau + t' < (h' + \gamma')\tau + (t' + 2) < h\tau + t < (h' + \beta')\tau + (t' + 1) < (h + \alpha)\tau + t$
We consider all possible W3RTZ cases used in the proof

Case2a : W3RTZ Case0
This W3RTZ case does not create a valid W5RTZ and can be ignored

Case2b : W3RTZ Case1a There are 3 possible vector summation for this W3RTZ case is

Case2b-a

$$\begin{array}{c}(\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_3 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1'') \\ \hline (\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1'')\end{array}$$

For Case2b-a, the derived vector is $(\boldsymbol{\phi}_2' \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_1' \ (\boldsymbol{\phi}_2')_{(h-h'+\beta'-1)} \ \boldsymbol{\phi}_2'' \ (\boldsymbol{\phi}_2)_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_1'')$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2(h-h') + 2\gamma' - 2 + 2 + 2(h'-h) + 2\beta' - 2 + 3 + 2(h-h') + 2\alpha - 2\beta - 2 + 1 \\ &= 2(h-h') + 2(\alpha + \gamma') \end{aligned}$$

Case2b-b

$$\begin{array}{c}(\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2''\boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2') \\ \hline (\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_2\boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2')\end{array}$$

For Case2b-b, the derived vector is $(\boldsymbol{\phi}_2' \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_2' \ (\mathbf{0}_3)_{(h-h'+\beta'-1)} \ \boldsymbol{\phi}_2 \ (\boldsymbol{\phi}_2'')_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_2')$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2(h-h') + 2\gamma' - 2 + 2 + 0(h'-h) + 0\beta' - 0 + 4 + 2(h-h') + 2\alpha - 2\beta - 2 + 2 \\ &= 4(h-h'+1) + 2(\alpha - \beta' + \gamma') \end{aligned}$$

Case2b-c

$$\begin{array}{c}(\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_3) \\ \hline (\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_1'' \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \mathbf{0}_3 \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_3)\end{array}$$

For Case2b-c, the derived vector is $(\boldsymbol{\phi}_2' \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_1'' \ (\boldsymbol{\phi}_2)_{(h-h'+\beta'-1)} \ \mathbf{0}_3 \ (\boldsymbol{\phi}_2')_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_3)$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2(h-h') + 2\gamma' - 2 + 2 + 2(h'-h) + 2\beta' - 2 + 1 + 2(h-h') + 2\alpha - 2\beta - 2 + 3 \\ &= 2(h-h') + 2(\alpha + \gamma') \end{aligned}$$

□

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