CHAPTER THREE

BUILDING BLOCKS OF CLASSIC CODING SCHEMES

3.1 CHAPTER OVERVIEW

CLASSIC block and convolutional coding scheme encoder building blocks are the focus of the first part of this chapter. Topics covered include the mathematical descriptions and definitions of several important characterisation parameters for binary convolutional codes, binary and non-binary linear block codes, interleavers and code puncturers. Both FIR and IIR type binary convolutional codes are investigated. Classic binary linear block code families described in this chapter include Hamming and BCH linear block codes, whereas RS block codes are considered in the discussion on non-binary linear block codes.

The second part of this chapter revolves around the decoder building blocks encountered in classic block and convolutional coding schemes. Since the basic ML and MAP decoder structures and algorithms, associated with classic block and convolutional codes, are readily available in the literature, such algorithms are not described in detail in this study. However, attention is given to the construction of binary convolutional code trellises. The remainder of this part of the chapter details the inner workings of de-interleavers and code de-puncturers. The chapter is concluded with a short discussion on the concept of CSI estimation, as well as several valuable references to interesting CSI estimation techniques.

3.2 ENCODER BUILDING BLOCKS

3.2.1 BINARY CONVOLUTIONAL CODES

This subsection is concerned with the basic theory of binary convolutional codes. Following a concise mathematical description of convolutional codes, attention is given to the FIR NSC and IIR RSC classes of convolutional codes.

3.2.1.1 MATHEMATICAL DESCRIPTION OF BINARY CONVOLUTIONAL CODES

A rate $R_c = k/n$ binary convolutional code encoder is essentially a finite state linear device, consisting of k separate shift registers (one for each input bit), that accepts k-tupple binary inputs and

generates n-tupple binary outputs [87]. The linearity property of such an encoder refers to the fact that a linear combination, in Galois field $GF(2)$, of a set of binary data blocks, used as input, results in a linear combination, in $GF(2)$, of the binary output code blocks, generated for each of the input blocks [47].

When describing convolutional codes, it is convenient to relate the encoder output to the encoder input by means of a generator matrix $G_{CC}(D)$ [87]: Let the i^{th} length-k sequence contained within the m^{th} vector of input bits into the encoder, and the i^{th} length-n sequence contained within the m^{th} vector of output bits out of the encoder, be denoted by $\overline{d}_{m,i} = \{d_{m,i,0}, d_{m,i,1}, ..., d_{m,i,k-1}\}$ and $\overline{c}_{m,i} = \{c_{m,i,0}, c_{m,i,1}, ..., c_{m,i,n-1}\}$, respectively. Using the D-transform [87], the stream of encoder inputs can be represented by the k-dimensional vector sequence $\overline{d}_m(D)$, given by:

$$\overline{d}_m(D) = \sum_i \overline{d}_{m,i} D^i \tag{3.1}$$

where D represents a single delay period of T_b [s]. Likewise, the stream of encoder outputs can be represented by the n-dimensional vector sequence $\overline{c}_m(D)$, given by:

$$\overline{c}_m(D) = \sum_i \overline{c}_{m,i} D^i \tag{3.2}$$

The generator matrix $G_{CC}(D)$ of the encoder is then the $k \times n$ matrix that satisfies the following relationship [87]:

$$\overline{c}_m(D) = \overline{d}_m(D).G_{CC}(D) \tag{3.3}$$

where the multiplication is carried out over $GF(2)$. In general, the form of the generator matrix is as follows [87]:

$$G_{CC}(D) = \begin{bmatrix} g_{0,0}(D) & g_{0,1}(D) & \dots & g_{0,n-1}(D) \\ g_{1,0}(D) & g_{1,1}(D) & \dots & g_{1,n-1}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,1}(D) & g_{k-1,1}(D) & \dots & g_{k-1,n-1}(D) \end{bmatrix} \tag{3.4}$$

where $g_{a,b}(D)$ is the generator polynomial describing the positions of the a^{th} input bit's shift register that must be linearly combined in $GF(2)$ to contribute to the b^{th} output bit.

Directly related to the generator matrix of a convolutional code, is its parity check matrix $H_{CC}(D)$. It is defined as an $(n-k) \times n$ matrix that, for an arbitrary code word vector sequence $\overline{c}_m(D)$, generated using the encoder's generator matrix $G_{CC}(D)$, satisfies the condition $\overline{c}_m(D).H_{CC}^T(D) = \overline{0}$ in $GF(2)$ [47], with $H_{CC}^T(D)$ the transpose of $H_{CC}(D)$.

Convolutional code encoders are classified as FIR or IIR type encoders [87]. The aforemention class of binary convolutional code encoders generate their outputs using only linear combinations of current and previous inputs. Thus, the generator polynomials of FIR type convolutional codes has the general form [87]:

$$g_{a,b}(D) = \sum_{j=0}^{\upsilon_a - 1} g_{a,b,j} D^j \tag{3.5}$$

where υ_a represents the total number of memory elements in the shift register used in conjunction with the a^{th} message word bit in $\overline{d}_{m,i}$. The variable $g_{a,b,j}$, which can take on values from the alphabet $\{0, 1\}$, indicates the presence or absence of a tap connecting the j^{th} memory element of the a^{th} shift register to the b^{th} output.

In the case of IIR type convolutional code encoders, not only is the current and previous inputs used to generate the current outputs, but also previous outputs. As such, the generator polynomials of IIR convolutional code encoders are rational functions in D [87].

3.2.1.2 IMPORTANT BINARY CONVOLUTIONAL CODE PARAMETERS AND DEFINITIONS

The following parameters and definitions are vital to the understanding of convolutional codes and their encoders:

1. **Hamming Distance:** The Hamming distance (see *Eq.* (4.29) in *Section* 4.4.2.1) between two separate encoder output vector sequences, $\overline{c}_m^1(D)$ and $\overline{c}_m^2(D)$, which is denoted by $d_H\left(\overline{c}_m^1(D), \overline{c}_m^2(D)\right)$, is defined as the number of bit positions in which they differ [47].

2. **Hamming Weight:** The Hamming weight $w_H\left(\overline{c}_m(D)\right)$ of an encoder vector output sequence $\overline{c}_m(D)$ is defined as the Hamming distance between $\overline{c}_m(D)$ and the all-zero vector sequence $\overline{0}$, i.e. $w_H\left(\overline{c}_m(D)\right) \triangleq d_H\left(\overline{c}_m(D), \overline{0}\right)$ [47].

3. **Constraint Length:** The constraint length of a rate $R_c = k/n$ convolutional code encoder is the number of delay elements used in its realisation. If the number of delay elements employed in the a^{th} input's shift register is denoted by υ_a, the constraint length of the encoder is given by [87]:

$$\upsilon = \sum_{a=0}^{k-1} \upsilon_a \tag{3.6}$$

   This parameter is the most important measure of the convolutional code's trellis complexity, since the number of states in the trellis of a binary convolutional code, with a constraint length of υ, is 2^{υ} (see *Section* 3.3.1.1).

4. **Minimal Encoders:** On closer inspection, it should be apparent that there might exist several encoder structures, each with its own memory (shift register) and tap configuration that might satisfy *Eq.* (3.3). However, it can be shown that there exists a subset of encoders, having identical state diagrams [47], which utilises a minimum number of memory elements to generate the convolutional code. Such encoders are called *minimal encoders* [87]. All the convolutional code encoders considered in this study are minimal encoders.

5. **Non-systematic Encoders:** At a certain encoding instance i of the m^{th} encoder input vector, the encoder input data stream of a non-systematic convolutional code do not form a substream of the encoder output data stream [87]. Thus, the encoder outputs bits consist solely of parity bits, i.e. $c_{m,i,a} = \text{v}_{m,i,a}$ for $a = 0, 1, ..., n-1$.

6. **Systematic Encoders:** A systematic convolutional code is one for which, at a certain encoding instance i of the m^{th} encoder input vector, the encoder input data stream forms a substream of the encoder output data stream [87]. The convention used throughout this study is that encoder output bits 0 to $k-1$ are the systematic bits, i.e. $c_{m,i,a} = d_{m,i,a}$ for $a = 0, 1, ..., k-1$, and outputs bits k to $n-1$ are the parity bits, i.e. $c_{m,i,a} = \text{v}_{m,i,a}$ for $a = k, k+1, ..., n-1$.

7. **Minimum Free Distance:** The minimum free distance d_{free} of a binary convolutional code is defined as [47]:

$$d_{free} \triangleq \min_{d_m^1(D) \neq d_m^2(D)} d_H\left(\overline{c}_m^1(D), \overline{c}_m^2(D)\right) \tag{3.7}$$

   where $\overline{c}_m^1(D) = \overline{d}_m^1(D).G_{CC}(D)$ and $\overline{c}_m^2(D) = \overline{d}_m^2(D).G_{CC}(D)$ in $GF(2)$. Essentially, d_{free} is a measure of how good a convolutional code is: The larger d_{free}, the better a code's performance,

i.e. more code bits must be in error in order for one code word to be mistaken for another by the decoder. Determining the structure of an optimal convolutional code encoder with a preset code rate and number of states in its trellis, involves an exhaustive search through all possible minimal encoders capable of generating the code, finally selecting the code with the largest d_{free}. Although not of crucial importance to this study, it is worth mentioning that d_{free} can be determined effortlessly from the code's *transfer function*, which in turn is determined from the encoder's *state diagram* [47].

8. **Asymptotic Coding Gain in AWGN Channel Conditions:** The asymptotic coding gain for binary convolutional codes, operating in AWGN channel conditions, decoded using soft decision ML decoding and employing QPSK modulation with coherent demodulation, is upper bounded as follows [47]:

$$CG_{CC}^{soft} \leq 10\log_{10}(R_c.d_{free}) \qquad \text{[dB]} \tag{3.8}$$

If hard decision decoding is employed, a 2 dB degradation in BER performance can be expected when compared to soft decision decoding, resulting in the following upper bound [47]:

$$CG_{CC}^{hard} \leq 10\log_{10}(R_c.d_{free}) - 2 \qquad \text{[dB]} \tag{3.9}$$

3.2.1.3 TYPES OF BINARY CONVOLUTIONAL CODES

Discussed in the following subsections are the two main types of binary convolutional codes, namely NSC and RSC codes. Although only NSC codes are used in classic coding schemes employing convolutional codes (due to the fact that RSC codes exhibit inferior BER performances at low values of E_b/N_0 when compared to NSC codes), both classes have found application as CCs in recently proposed iteratively decoded concatenated coding schemes.

3.2.1.3.1 Finite Impulse Response Non-Systematic Convolutional Codes

Although both IIR and FIR NSC codes can be constructed, FIR type NSC codes have proven to be a more attractive solution in classic convolutional coded systems, as well as recent *Serial Concatenated Convolutional Code* (SCCC) schemes [27,29]. As such, this study only concerns itself with FIR NSC codes.

Tables with the generator polynomials of the most optimal binary convolutional code encoders, along with their associated d_{free} values, have been extensively documented in the literature. As an example, *Fig.* 3.1 shows the encoder structure of a minimal 8-state, rate $R_c = 1/2$, $\upsilon = 3$, FIR NSC code, taken from [47]. Using the generator matrix encoder description method detailed in *Section* 3.2.1.1, the generator matrix defining this encoder is:

$$G_{CC}(D) = \left[\begin{array}{cc} 1 + D + D^3 & 1 + D + D^2 + D^3 \end{array}\right] \tag{3.10}$$

It has been shown [47] that the NSC code generated by this generator matrix has a free distance of $d_{free} = 6$. This specific NSC code is used extensively in the simulations detailed in *Chapter* 6.

3.2.1.3.2 Recursive Systematic Convolutional Codes

The importance for the constituent encoders of *Parallel Concatenated Convolutional Code* (PCCC) encoding schemes to be both systematic for decoding simplicity, and recursive in order to maximise the *interleaver gain*, is now well recognised in the literature [19, 88]. As such, a class of systematic IIR codes have been proposed [89] for the building blocks of PCCC encoders. These codes are com-

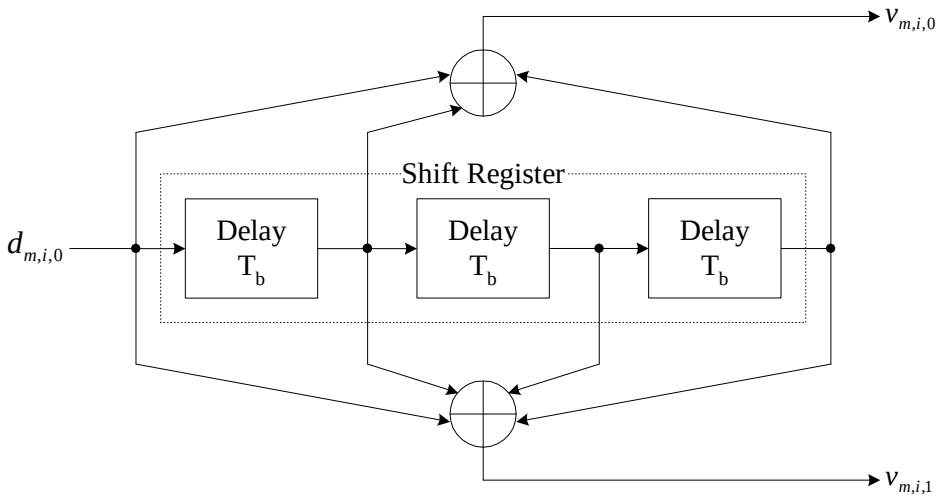

Figure 3.1: Optimal 8-State, Rate $R_c = 1/2$ NSC Code Encoder

monly referred to as RSC codes.

The RSC CC encoders used in a PCCC coding scheme need not be identical with regard to their constraint lengths or rates. When designing a PCCC encoder, the goal is to choose the best component codes by maximising the effective free distance [90] of the PCCC. At large values of E_b/N_0, this is tantamount to maximising the minimum weight code word [91, 92]. However, at low values of E_b/N_0 (the region of greatest interest) optimising the weight distribution of the code words is more important than maximising the minimum weight [91].

Listed in *Appendix A* are the encoder parameters of one of the most extensive sets of optimal RSC encoders, as determined by *Benedetto*, *Garello* and *Montorsi* through exhaustive searches [88]. Also specified for each of the encoders listed, is the minimum free distance d_{free}. The construction of the optimal 8-state, $\upsilon = 3$, rate $R_c = 2/3$ RSC encoder (see *Fig.* A.2) in *Section* A.3 illustrates how the listed encoder parameters are interpreted. The generator matrix of this encoder, which is employed in several of the simulations discussed in *Chapter* 6, is given by:

$$G_{CC}(D) = \begin{bmatrix} 1 & 0 & \frac{1+D^2+D^3}{1+D+D^3} \\ 0 & 1 & \frac{1+D+D^2}{1+D+D^3} \end{bmatrix} \tag{3.11}$$

According to *Table* A.5, the minimum free distance of this code is $d_{free} = 4$.

Another example of an RSC code encoder is shown in *Fig.* 3.2. In this figure, the encoder structure for an optimal 8-state, $\upsilon = 3$, rate $R_c = 1/2$ RSC code is depicted. From *Table* A.3 in *Appendix A* it follows that this code has a minimum free distance of $d_{free} = 6$ and is defined by the following generator matrix:

$$G_{CC}(D) = \begin{bmatrix} 1 & \frac{1+D+D^2+D^3}{1+D^2+D^3} \end{bmatrix} \tag{3.12}$$

3.2.2 LINEAR BLOCK CODES

The focus of this subsection falls on linear block codes. Following a concise mathematical description of general linear block codes, a number of important linear block code definitions and parameters are explained. A short description of the characteristics of binary Hamming, binary BCH and non-binary

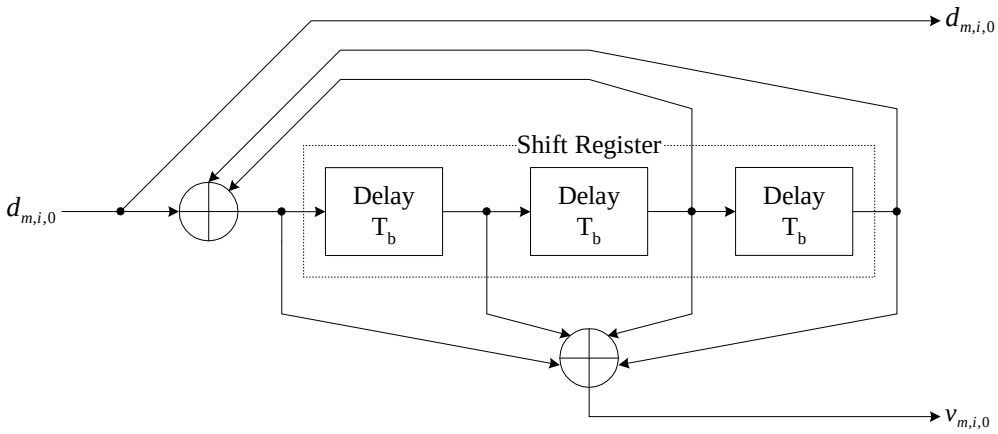

Figure 3.2: Optimal 8-State, Rate $R_c = 1/2$ RSC Code Encoder

RS block codes conclude the subsection.

3.2.2.1 MATHEMATICAL DESCRIPTION OF LINEAR BLOCK CODES

Consider an (n, k, d_{min}) linear block code with message and code word symbols from $GF\left(2^{\xi}\right)$. This code has $2^{\xi.k}$ unique n-symbol code words and a minimum Hamming distance (see *Section* 3.2.2.2) of d_{min} [symbols]. Suppose, at encoding instance m, the n-symbol code word vector $\overline{c}_m = \{c_{m,0}, c_{m,1}, c_{m,2}, ..., c_{m,n-1}\}$ is the output produced by the linear block code encoder, given the k-symbol input message word $\overline{d}_m = \{d_{m,0}, d_{m,1}, d_{m,2}, ..., d_{m,k-1}\}$. The encoding process performed by the linear block code encoder can be described by means of a set of n linear equations [47, 93, 94]:

$$c_{m,j} = d_{m,0}.g^e_{0,j} + d_{m,1}.g^e_{1,j} + ... + d_{m,k-1}.g^e_{k-1,j} \qquad \text{for } j = 0, 1, ..., n-1 \tag{3.13}$$

where the variables $g^e_{i,j}$, with $i = 0, 1, ..., k-1$ and $j = 0, 1, ..., n-1$, dictate the one-to-one relationship, specific to the type of linear block code and the code constraints, between $\overline{c}_m$ and $\overline{d}_m$. These variables can only take on values from $GF\left(2^{\xi}\right)$. Furthermore, the multiplication and addition operations of *Eq.* (3.13) and all subsequent equations are also performed in $GF\left(2^{\xi}\right)$. A more convenient method that can used to describe the encoding process, is as follows [47, 93, 94]:

$$\overline{c}_m = \overline{x}_m.G_{BC} \tag{3.14}$$

where G_{BC}, commonly referred to as the generator matrix of the block code, is a rank k size $k \times n$ matrix, given by:

$$G_{BC} = \begin{bmatrix} g^e_{0,0} & g^e_{0,1} & \cdots & g^e_{0,n-1} \\ g^e_{1,0} & g^e_{1,1} & \cdots & g^e_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g^e_{k-1,0} & g^e_{k-1,1} & \cdots & g^e_{k-1,n-1} \end{bmatrix} \tag{3.15}$$

The linearity characteristic of a linear block code refers to the fact that the $GF\left(2^{\xi}\right)$ linear combination of two distinct code words $\overline{c}^1_m$ and $\overline{c}^2_m$, generated by G_{BC} for the respective message words $\overline{d}^1_m$ and $\overline{d}^2_m$, is equal to the encoder output code word $\overline{c}^3_m$ for the input message word $\overline{d}^3_m = \overline{d}^1_m + \overline{d}^2_m$. Thus, any of the $2^{\xi.k}$ code words in the linear block code can be constructed by linearly combining several of the other code words in $GF\left(2^{\xi}\right)$ [47, 93, 94].

Closely related to the generator matrix G_{BC}, is a linear block code's $(n-k) \times n$ parity check matrix H_{BC}, defined by the following relationship [47, 93, 94]:

$$G_{BC}.H_{BC}^{T} = \overline{0} \tag{3.16}$$

where $\overline{0}$ denotes an all-zero $k \times (n-k)$ matrix and H_{BC}^{T} is the transpose of H_{BC}. With the linearity property in mind, it can be shown that all of the $2^{\xi \cdot k}$ code words in the linear block code is orthogonal to every row of the parity check matrix H_{BC}. As such, the parity check matrix is used extensively in algebraic linear block code decoding techniques [47, 93, 94], such as syndrome decoding, in order to isolate and possibly correct errors in corrupted code words.

3.2.2.2 IMPORTANT LINEAR BLOCK CODE PARAMETERS AND DEFINITIONS

Several parameters and definitions that are crucial for the understanding and characterisation of linear block codes are listed below. Since these parameters and definitions are valid for both binary and non-binary linear codes, all addition and multiplication operations present in this discussion are carried out in the Galois field over which the linear block code is defined.

1. **Hamming Distance:** Identical to the definition given for binary convolutional codes in *Section* 3.2.1.2, the Hamming distance between two block code encoder output code words $\overline{c}_m^1$ and $\overline{c}_m^2$, denoted by $d_H\left(\overline{c}_m^1, \overline{c}_m^2\right)$, is defined as the number of code word symbol positions in which they differ [47, 93, 94].

2. **Minimum Hamming Distance:** Analogous to the d_{free} of a binary convolutional code (see *Section* 3.2.1.2), the minimum Hamming distance of a linear block code, denoted by d_{min}, is the most salient measure of the error detection and correction capabilities of the code. Its mathematical definition is as follows [47, 93, 94]:

$$d_{min} \triangleq \min_{d_m^1 \neq d_m^2} d_H\left(\overline{c}_m^1, \overline{c}_m^2\right) \tag{3.17}$$

   where $\overline{c}_m^1 = \overline{d}_m^1.G$ and $\overline{c}_m^2 = \overline{d}_m^2.G$. If the minimum Hamming distance of a linear block code is known, it can be shown [47, 93, 94] that the number of code word symbol errors that is detectable by the code for a single code word, is $t_{detect} = d_{min} - 1$, whereas the number of correctable code word symbol errors is $t_{correct} = \left\lfloor \frac{1}{2}\left(d_{min} - 1\right) \right\rfloor$.

3. **Hamming Weight:** The Hamming weight $w_H\left(\overline{c}_m\right)$ of a linear block code's output code word $\overline{c}_m$ is defined as the Hamming distance between $\overline{c}_m$ and the all-zero code word $\overline{0}$, i.e. $w_H\left(\overline{c}_m\right) \triangleq d_H\left(\overline{c}_m, \overline{0}\right)$ [47, 93, 94].

4. **Non-systematic Linear Block Codes:** Identical to the definition for binary non-systematic convolutional codes, the message word $\overline{d}_m$ used as input into a non-systematic linear block code encoder at timing instance m, does not form a substream of the encoder output code word $\overline{c}_m$. Thus, the encoder output code word symbols consist solely of parity symbols, i.e. $c_{m,i} = \mathrm{v}_{m,i}$ for $i = 0, 1, 2, ..., n-1$.

5. **Systematic Linear Block Codes:** An (n, k, d_{min}) linear block code is said to be systematic if the length-n encoder output code word $\overline{c}_m$, generated at encoding instance m, contains a length-k substring that is an exact replica of the k-symbol encoder input message word $\overline{d}_m$. The convention adopted throughout this study for such linear block codes, is that the first k code word symbols of $\overline{c}_m$ are the systematic symbols, i.e. $c_{m,i} = d_{m,i}$ for $i = 0, 1, 2, ..., k-1$, and the last $n-k$ code word symbols are the parity symbols added by the encoder, i.e. $c_{m,i} = \mathrm{v}_{m,i}$ for $i = k, k+1, ..., n-1$.

Hence, the generator matrix of such a systematic linear block code has the following general form [47, 93, 94]:

$$G_{BC} = [I_k|P] = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & P^s_{0,0} & P^s_{0,1} & \dots & P^s_{0,n-k-1} \\ 0 & 1 & \dots & 0 & P^s_{1,0} & P^s_{1,1} & \dots & P^s_{1,n-k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & P^s_{k-1,0} & P^s_{k-1,1} & \dots & P^s_{k-1,n-k-1} \end{array}\right] \tag{3.18}$$

where I_k is the $k \times k$ identity matrix and P^s is a $k \times (n-k)$ matrix that determines the $n-k$ parity symbols added to each code word during encoding. An important characteristic of linear block codes, is that any non-systematic generator matrix, given by *Eq.* (3.15), can be reduced to the systematic form of *Eq.* (3.18) by means of a number of row operations and column permutations [47] (using Gaussian elimination is a popular approach).

Recalling that addition and subtraction are equivalent in $GF(2)$, it can be shown that the parity check matrix of a binary linear systematic block code is easily determined as follows [47, 93, 94]:

$$H_{BC} = \left[(P^s)^T | I_{n-k}\right] \tag{3.19}$$

where $(P^s)^T$ represents the transpose of P^s and I_{n-k} the $(n-k) \times (n-k)$ identity matrix.

6. **Weight Enumerating Function:** The *Weight Enumerating Function* (WEF) of an (n, k, d_{min}) linear block code is a compact method to describe its weight distribution. It is defined as follows [47, 93, 94]:

$$A(Z) = \sum_{w=0}^{n} A_w Z^w \tag{3.20}$$

where A_w is the number of code words in the block code that has a Hamming weight of w. The WEF can be used to compute the exact expression of the probability of undetected errors, as well as upper bounds on the word error probability.

7. **Input-Output Weight Enumerating Function:** The *Input-Output Weight Enumerating Function* (IOWEF) of an (n, k, d_{min}) linear block code, also sometimes referred to as the input-redundancy weight enumerating function, splits each term in the WEF into the separate contributions of the parity and message word symbols to the total Hamming weight of each code word. It is defined as follows [27, 63, 66, 87]:

$$A(W, Z) = \sum_{w=0}^{k} \sum_{h=0}^{n} A_{w,h} W^w Z^h \tag{3.21}$$

where $A_{w,h}$ is the number of code words in the block code that has a Hamming weight of h, generated by message words with a Hamming weight of w.

8. **Cyclic Linear Block Codes and Their Generator Polynomials:** Cyclic linear block codes have the property that all possible cyclic shifts of the elements of one code word results in another valid code word [47, 93, 94]. When describing these block codes, it is convenient to express a message word $\overline{d}_m = \{d_{m,0}, d_{m,1}, ..., d_{m,k-1}\}$ and its associate code word $\overline{c}_m = \{c_{m,0}, c_{m,1}, ..., c_{m,n-1}\}$ in terms of a degree $k-1$ message polynomial $d_m(p) = d_{m,k-1}.p^{k-1} + d_{m,k-2}.p^{k-2} + ... + d_{m,0}$ and a degree $n-1$ code word polynomial $c_m(p) = c_{m,n-1}.p^{n-1} + c_{m,n-2}.p^{n-2} + ... + c_{m,0}$, respectively. Using this notation, the cyclic nature of the code can be easily verified: If $\overline{c}^1_m$ is a code word from the cyclic linear block code, $\overline{c}^2_m$ will also be a valid code word, given that the

following condition is satisfied [47, 93, 94]:

$$\overline{c}_m^2 = p^i.\overline{c}_m^1 \mod (p^n + 1) \tag{3.22}$$

where i can take on any integer value.

Generation of the code word polynomial $c_m(p)$, given the message polynomial $d_m(p)$, can be described using a degree $n-k$ generator polynomial $g_{BC}(p) = g_{n-k}^c.p^{n-k} + g_{n-k-1}^c.p^{n-k-1} + ... + g_0^c$. The encoding process is as follows:

1. *Non-systematic Encoding* - If a cyclic linear block code has to generate non-systematic code words, the encoding process of message polynomial $d_m(p)$ to code word polynomial $c_m(p)$ is as follows [47]:
$$c_m(p) = d_m(p).g_{BC}(p) \tag{3.23}$$
2. *Systematic Encoding* - The systematic encoding of a message polynomial $d_m(p)$ by a cyclic linear block code encoder to give the code word polynomial $c_m(p)$, entails the following [47]:
   (a) Multiply the message polynomial $d_m(p)$ by p^{n-k}.
   (b) Obtain the remainder polynomial $r_m(p)$ from the division of $p^{n-k}.x_m(p)$ by $g_{BC}(p)$.
   (c) Construct the systematic code word polynomial as follows:
$$c_m(p) = p^{n-k}.d_m(p) + r_m(p) \tag{3.24}$$

The construction of a non-systematic generator matrix G_{BC} (see *Section* 3.2.2.1) for a cyclic linear block code from its generator polynomial, is easily accomplished by firstly setting $g_{i,(n-k-j+i)}^e = g_j^c$ for $i = 0, 1, ..., k-1$ and $j = 0, 1, 2, ..., n-k$. All the remaining elements of G_{BC} are then set to zero [47].

9. **Coding Gain in AWGN Channel Conditions:** The coding gain of an (n, k, d_{min}) linear block code, operating in AWGN channel conditions, decoded using soft decision ML decoding and employing QPSK modulation with coherent demodulation, is upper bounded as follows [47]:
$$CG_{BC}^{soft} \leq 10\log_{10}\left(R_c.d_{min} - k.\frac{N_0}{E_b}.\ln(2)\right) \quad \text{[dB]} \tag{3.25}$$
Typically a 2 dB degradation in BER performance can be expected if hard decision decoding is employed. Hence, the following upper bound can be calculated for the hard decision decoding approach [47]:
$$CG_{BC}^{hard} \leq 10\log_{10}\left(R_c.d_{min} - k.\frac{N_0}{E_b}.\ln(2)\right) - 2 \quad \text{[dB]} \tag{3.26}$$

3.2.2.3 LINEAR BLOCK CODES OF IMPORTANCE FOR THIS STUDY

The next two subsections describe the main characteristics of binary Hamming and BCH block codes, respectively. This is then followed by a subsection that gives a brief description of non-binary RS block codes.

3.2.2.3.1 Binary Hamming Block Codes

R. W. Hamming first presented the well-known class of linear Hamming block codes in 1950 [95]. Since then, it has been shown that these single error correcting systematic or non-systematic block

codes can be classified as cyclic linear block codes (see *Section* 3.2.2.2). Although both binary and non-binary Hamming block codes can be constructed, this study is limited to the binary class of Hamming block codes, characterised by the following properties [47]:

$$(n, k, d_{min}) = (2^a - 1, 2^a - 1 - a, 3) \tag{3.27}$$

where a is a positive integer.

Another property of an $(n, k, 3)$ binary Hamming block code, frequently used in its construction, is that the n columns of its parity check matrix H_{BC} consists of all possible binary vectors with $n - k = a$ elements, except the all-zero vector.

3.2.2.3.2 Binary Bose-Chaudhuri-Hocquenghem Block Codes

BCH block codes, discovered independently by *Hocquenghem* in 1959 [96], and *Bose* and *Ray-Chaudhuri* in 1960 [97, 98], comprise of a large class of cyclic linear block codes defined over binary and non-binary symbol alphabets. Although this subsection only covers binary BCH block codes, the popular class of RS block codes, which is a subclass of non-binary BCH block codes, is briefly described in *Section* 3.2.2.3.3.

Binary BCH codes are constructed in compliance with the following code parameter restrictions: The number of code word bits per code word is given by [47]:

$$n = 2^a - 1 \tag{3.28}$$

where $a \geq 3$ is an integer value. If the number of correctable bit errors per code word is $t_{correct}$, the number of parity bits added to each message word during encoding is bounded as follows [47]:

$$n - k =\leq a.t_{correct} \tag{3.29}$$

resulting in the following minimum Hamming distance [47]:

$$d_{min} = 2.t_{correct} + 1 \tag{3.30}$$

The generator polynomial $g_{BC}(p)$ for such a binary BCH code can be constructed from the factors of $p^{2^a-1} + 1$. An extensive list of binary BCH block code generator polynomials for $2 \leq a \leq 34$ is presented in [99].

3.2.2.3.3 Non-binary Reed-Solomon Block Codes

The subclass of maximum distance non-binary BCH block codes, commonly known as RS block codes [14], is the focus of this subsection. The subsection starts by outlining a number of basic RS block code parameters and characteristics. This is then followed by a section that presents a short discussion on classic cyclic encoding.

3.2.2.3.3.1 Parameters and Characteristics of Reed-Solomon Block Codes

Consider an (n, k, d_{min}) RS block code with message and code word symbols from the extended binary Galois field $GF\left(2^{\xi}\right)$, where $\xi > 1$. Assuming that the RS block code has to correct $t_{correct}$ symbol errors, it can be characterised by the following set of parameters: The number of code word

and parity symbols per code word are given by [14, 93]:

$$n = 2^{\xi} - 1 \tag{3.31}$$

and

$$n - k = 2.t_{correct} \tag{3.32}$$

respectively. Measured in $GF\left(2^{\xi}\right)$ symbols, the minimum Hamming distance of the RS block code is [14]:

$$d_{min} = 2.t_{correct} + 1 \tag{3.33}$$

An important characteristic of RS block codes is that these codes are maximum distance codes, i.e. for a given n and k, there is no other linear block code that has a larger d_{min} than an RS code [14, 47, 93].

Since encoding takes place in $GF\left(2^{\xi}\right)$, message and code word symbols will consist of length-ξ binary streams, uniquely defined for each of the 2^{ξ} symbols in $GF\left(2^{\xi}\right)$. Thus, the total number of bits used per message and code word are $n_{bits} = n.\xi$ and $k_{bits} = k.\xi$, respectively.

3.2.2.3.3.2 Classic Encoding of Reed-Solomon Block Codes

Recall from *Section* 3.2.2.3.2 that BCH block codes are cyclic linear block codes. Consequently, an (n, k, d_{min}) RS block code, with message and code word symbols from $GF\left(2^{\xi}\right)$, is also a cyclic linear block code, which can be described by means of a degree-$(n-k)$ generator polynomial with generator coefficients from $GF\left(2^{\xi}\right)$ [93]. Assuming the RS block code can correct $t_{correct}$ symbol errors, the general form of this generator polynomial is as follows [14]:

$$g_{BC}\left(p\right) = \prod_{i=1}^{2.t_{correct}} \left(p + \varphi^{i}\right) \tag{3.34}$$

where φ is the primitive element of $GF\left(2^{\xi}\right)$. Note that this element satisfies the condition $g\left(\varphi^{i}\right) = 0$ for any integer i. For example, consider the $t_{correct} = 1$ symbol correcting RS $(7, 5, 3)$ code, operating in $GF\left(2^{3}\right)$. Since $GF\left(2^{3}\right)$ is defined by the irreducible (primitive) polynomial $g_{ip}(p) = 1 + p + p^{3}$, it follows from *Eq.* (3.34) that the generator polynomial of the RS $(7, 5, 3)$ code is the following:

$$g_{BC}\left(p\right) = \prod_{i=1}^{2} \left(p + \varphi^{i}\right) = (p + \varphi)(p + \varphi^{2}) = \varphi^{3} + \varphi^{4}p + p^{2} \tag{3.35}$$

The generator polynomial can now be used to generate either systematic or non-systematic code words, as described in *Section* 3.2.2.2.

3.2.3 INTERLEAVERS

An interleaver can be described as a simple single input, single output device that takes symbols from a fixed alphabet as input and produces an identical set of symbols at the output in an altered temporal order [87]. Thus, the basic function of an interleaver is to effectively shuffle the order of a sequence of symbols. In traditional applications, interleaving was used to ”randomise” the locations of errors caused by bursty (correlative) channels, which in turn improves the performance of classic block and convolutional coding schemes designed and optimised for non-correlative channels, such as the AWGN channel (see *Section* 2.2). With iteratively decoded concatenated coding schemes, however, interleavers are used mainly to decrease the correlation between the information encoded by the different CCs, thereby improving the distance properties of the resultant concatenated code.

3.2.3.1 MATHEMATICAL DESCRIPTION OF INTERLEAVERS

Let the m^{th} length-N sequence of symbols used as input to an interleaver π be denoted by $\overline{\mu}_m^{in} = \{\mu_{m,0}^{in}, \mu_{m,1}^{in}, ..., \mu_{m,(N-2)}^{in}, \mu_{m,(N-1)}^{in}\}$. The interleaving of $\overline{\mu}_m^{in}$ by π can be described as follows:

$$\pi\left(\overline{\mu}_m^{in}\right) = \overline{\mu}_m^{out} = \{\mu_{m,0}^{out}, \mu_{m,1}^{out}, ..., \mu_{m,(N-2)}^{out}, \mu_{m,(N-1)}^{out}\} \tag{3.36}$$

where $\overline{\mu}_m^{out}$ represents the N-symbol/sample output of the interleaver π. This output can also be written as:

$$\overline{\mu}_m^{out} = \{\mu_{m,0}^{out}, \mu_{m,1}^{out}, ..., \mu_{m,(N-2)}^{out}, \mu_{m,(N-1)}^{out}\} = \{\mu_{m,\Pi(0)}^{out}, \mu_{m,\Pi(1)}^{out}, ..., \mu_{m,\Pi(N-2)}^{out}, \mu_{m,\Pi(N-1)}^{out}\} \tag{3.37}$$

where $\Pi(i)$, with i any integer value, is a function that describes the mapping of the interleaver output time indices to interleaver input time indices. Since interleaving can be considered as a periodic re-ordering of blocks of N symbols, the function $\Pi(i)$ describes a one-to-one mapping over the integers i modulo the period N. Thus, it follows that:

$$\Pi(i) - N = \pi(i - N) \qquad \text{for all } i \tag{3.38}$$

For example, the simple $N = 3$ interleaver described in [87] is defined by the following mapping function:

$$\Pi(i) = \begin{cases} i & \text{if } i \bmod 3 = 0 \\ i - 3 & \text{if } i \bmod 3 = 1 \\ i - 6 & \text{if } i \bmod 3 = 2 \end{cases} \tag{3.39}$$

The interleaver mapping function can also be described in terms of a *fundamental permutation*, defined as follows:

$$\eth\left(\Pi\right) = \begin{pmatrix} 0 & 1 & \cdots & N-1 \\ \Pi(0) & \Pi(1) & \cdots & \Pi(N-1) \end{pmatrix} \tag{3.40}$$

For example, the fundamental permutation of the interleaver, defined by the mapping function of *Eq.* (3.39), is given by

$$\eth\left(\Pi\right) = \begin{pmatrix} 0 & 1 & 2 \\ 0 & -2 & -4 \end{pmatrix} \tag{3.41}$$

The remaining values of the interleaver mapping function, spanning all integer values of i, are obtained by combining the fundamental permutation, given by *Eq.* (3.40), and the periodicity condition, given by *Eq.* (3.38). Shown in *Table* 3.1 is the interleaver mapping function for the example interleaver of *Eq.* (3.39), calculated for $i = -1, 0, ..., 5$. To further the understanding of interleaving,

Table 3.1: Mapping of the Simple $N = 3$ Interleaver, Described by *Eq.* (3.39)

i	-1	0	1	2	3	4	5
$\Pi(i)$	-7	0	-2	-4	3	1	-1

the stream of interleaver input sequences can be represented by the N-dimensional D-transform [87] vector sequence $\overline{\mu}^{in}(D)$, given by:

$$\overline{\mu}^{in}(D) = \sum_m \overline{\mu}_m^{in} D^m \tag{3.42}$$

where D represents one interleaver delay period of N symbols. Using this representation for the interleaver input stream, the interleaver output stream can be described as follows:

$$\overline{\mu}^{out}(D) = \overline{\mu}^{in}(D) \cdot G_{\pi}(D) \tag{3.43}$$

where $G_{\pi}(D)$, referred to as the *generator matrix* of the interleaver, is an $N \times N$ non-singular matrix with the following restrictions:

1. Only one entry in each row/column can be non-zero to ensure a one-to-one mapping.
2. Non-zero entries are of the form D^i, where i is an integer.

For example, the generator matrix associated with the interleaver mapping, given by *Eq.* (3.39) and illustrated in *Table* 3.1, is as follows:

$$G_{\pi}(D) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & D^1 & 0 \\ 0 & 0 & D^2 \end{bmatrix} \tag{3.44}$$

From this generator matrix it is clear that the implementation of the interleaver requires the use of delay elements. In general, such interleavers are called *convolutional interleavers*, since they are constructed using shift registers, not unlike convolutional code encoders (see *Section* 3.2.1.1). More information concerning the structure and characteristics of these interleavers can be found in [87]. Interleavers that do not require the use of delay elements are referred to as *block interleavers*. This type of interleaver is discussed in more detail in *Section* 3.2.3.3.

3.2.3.2 INTERLEAVER PARAMETERS

The following parameters are often encountered in the study and characterisation of interleavers and de-interleavers:

1. **Interleaver Delay:** The delay of an interleaver is defined as the total delay introduced into a system by first interleaving and then de-interleaving a block of N symbols [87].
2. **Interleaver Causality:** A causal interleaver has the property that all elements in $G_{\pi}(D)$ are of the form D^i, with $i \geq 0$ for elements on, or under the diagonal of $G_{\pi}(D)$, and $i \geq 1$ for elements above the diagonal [87]. Therefore, the interleaver defined by *Eq.* (3.39) is causal.
3. **Interleaver Memory:** The memory of an interleaver is defined as the minimum number of memory elements required to implement a causal version thereof [87]. It is easily calculated by summing the absolute values of the exponents of the D elements in $G_{\pi}(D)$.
4. **Interleaver Spreading Factor:** If an interleaver has the spreading factor (M_{spread}, t_{burst}), it indicates that the individual symbols in a burst of a length smaller than t_{burst} symbols at the input of the interleaver are separated into distinct blocks of a length greater than or equal to M_{spread} at the output [87].
5. **Interleaver Dispersion:** The dispersion of an interleaver can be used to study its "randomness" [87]. It is calculated by determining the number of unique *displacement vectors* of the interleaver, normalised with respect to $N \cdot (N-1)/2$. The interested reader is referred to [87] for a discussion on the calculation of interleaver displacement vectors.

3.2.3.3 BLOCK INTERLEAVER STRUCTURES

It has been shown [87] that block interleavers, i.e. interleavers with generator matrices $G_\pi(D)$ having elements with only exponents of zero, are better suited than convolutional interleavers for both classic block and iteratively decoded concatenated coding techniques. Therefore, only block interleavers are of importance for this study.

Several types of block interleaver structures are available to the communications engineer. *Appendix C* considers several popular deterministic and random block interleavers that have attracted the attention of classic block and concatenated code designers over recent years. The deterministic interleaver structures discussed comprise of classic block interleavers, *Berrou-Glavieux* interleavers and JPL interleavers, whereas the random interleavers of interest are PN generator interleavers, random number generator interleavers and s-random interleavers. A description of the *uniform interleaver* [100, 101], a probabilistic device frequently encounter in the mathematical derivation of BER performance bounds for concatenated codes, concludes the appendix.

3.2.4 CODE PUNCTURERS

The price of the performance gains obtained by employing low rate codes in communication systems, is increased transmission bandwidths and/or lower data rates. Fortunately, by using a process called *code puncturing* [102–105], it is possible to maintain most of a code's error correcting capabilities, but increase the code rate, thereby decreasing the required transmission bandwidth. Consequently, this technique allows for the use of a single code over a wide variety of code rates with negligible performance losses. As such, it has become an indispensable component in the channel coding subsystems of numerous prevalent wireless communication standards. For example, it is used extensively in the voice, data and signalling channel coding schemes employed by GSM, as well as the 4 coding schemes (denoted CS-1 through CS-4) of *General Packet Radio Service* (GPRS).

Code puncturing is accomplished by deleting selected encoder output bits according to a chosen perforation pattern, also known as a *puncturing profile*. In general, the period-M_{punct} puncturing profile used to increase the code rate of a binary rate $R_c = k/n$ code, can be expressed by the following matrix:

$$\Upsilon = \begin{bmatrix} \Upsilon_{0,0} & \Upsilon_{0,1} & \cdots & \Upsilon_{0,M_{punct}-1} \\ \Upsilon_{1,0} & \Upsilon_{1,1} & \cdots & \Upsilon_{0,M_{punct}-1} \\ \vdots & \vdots & \ddots & \vdots \\ \Upsilon_{n-1,0} & \Upsilon_{n-1,0} & \cdots & \Upsilon_{n-1,M_{punct}-1} \end{bmatrix} \tag{3.45}$$

where the elements of Υ can only take on the values 0 and 1. The puncturing profile specifies that the j^{th} code bit of the i^{th} n-bit encoder output is deleted from the stream of coded bits to be transmitted, if $\Upsilon_{j,a} = 0$, where $a = i \bmod M_{punct}$. Alternatively, if $\Upsilon_{j,a} = 1$, the specific code bit is preserved. The code rate achieved after puncturing is easily calculated as follows:

$$R_p = \frac{k.M_{punct}}{\sum_{j=0}^{n-1} \sum_{a=0}^{M_{punct}-1} \Upsilon_{j,a}} \tag{3.46}$$

For example, assume a single rate $R_c = 1/3$ RSC code (see *Section* 3.2.1.3.2) has to be used in a communication system, but the required code rate is $1/2$. A further requirement is that the systematic output bits generated by the encoder may not be punctured. One possible period-1 puncturing approach is to permanently delete one of the two parity bits generate by the encoder for every input bit. However, this might lead to an unacceptable degradation in the code's performance. A more

attractive solution is to alternate the puncturing between the two parity bits. The following period-2 puncturing profile is one of two possible profiles that will implement the aforementioned puncturing requirements:

$$\Upsilon = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.47}$$

3.3 DECODER BUILDING BLOCKS

3.3.1 BINARY CONVOLUTIONAL CODE DECODERS

Since classic ML decoding techniques [106] have become cornerstone algorithms in convolutional code decoding, they are not repeated here. The unenlightened reader is referred to [106] for comprehensive explanations and examples of the application of the VA in the decoding of convolutional codes, and [107] for a discussion on the non-optimal *Fano* sequential decoding technique.

Most ML convolutional code decoding algorithms (such as the *sliding window* VA) and MAP convolutional code decoding algorithms (such as the BCJR algorithm [2]) make use of convolutional code *trellises*. The trellis of a rate $R_c = k/n$ binary convolutional code is essentially a time-indexed state diagram for the underlying binary circuit defined by the generator matrix $G_{CC}(D)$ [47]. As such, it contains all relevant information crucial for ML and MAP decoding algorithms. The following subsection describes the construction of binary block code trellises.

3.3.1.1 CONSTRUCTING THE TRELLIS OF A BINARY CONVOLUTIONAL CODE

Given that the constraint length of the convolutional code encoder is υ, the trellis of the code has 2^{υ} states, with trellis state l being defined as the decimal equivalent of the υ-bit binary number created by concatenating the encoder's memory elements' outputs at a certain encoding instance. The number of branches leaving or entering a state (node) in a binary convolutional code's trellis, is either 0 (in the fan-out section of the trellis) or 2^k [47].

An important characteristic of a depth-$M_{sections}$ convolutional code trellis, distinguishing it from a linear block code trellis (see *Section* 4.2), is that it can be constructed by concatenating $M_{section}$ identical trellis sections, where a trellis section is defined as a single depth trellis showing all possible state transitions of the convolutional code encoder under investigation [47]. Consequently, a convolutional code's trellis can be described as being "time-invariant", whereas a linear block code's trellis is "time-variant", since each trellis section is unique.

Construction of a single convolutional code trellis section involves determining the destination state and associated n-bit encoder output, given that the i^{th}, for $i = 1, 2, ..., 2^k$, possible k-bit encoder input is used with the encoder in an origin state l, for $l = 0, 1, ..., 2^{\upsilon} - 1$. The transition from an initial (origin) state to a destination state is indicated by the presence of a branch. Each branch has an associated n-bit branch weight or decoder input branch vector, as well as a k-bit decoder output branch vector. The decoder input and output branch vectors of the j^{th} branch leaving the l^{th} state at a trellis depth of i are denoted by $\overline{u}_{i,l}^{(j)}$ and $\overline{o}_{i,l}^{(j)}$, respectively.

In the graphical representation of a single trellis section, the decoder input and output sequences, associated with each branch, are usually indicated by means of a *"Decoder Output Sequence / Decoder Input Sequence"* (or equivalent *"Encoder Input Sequence / Encoder Output Sequence"*) label. *Fig.* 3.3 shows such a trellis section, obtained by following the foregoing procedure for the optimal

8-state, $\upsilon = 3$, rate $R_c = 1/2$ RSC code, defined by *Eq.* (3.12).

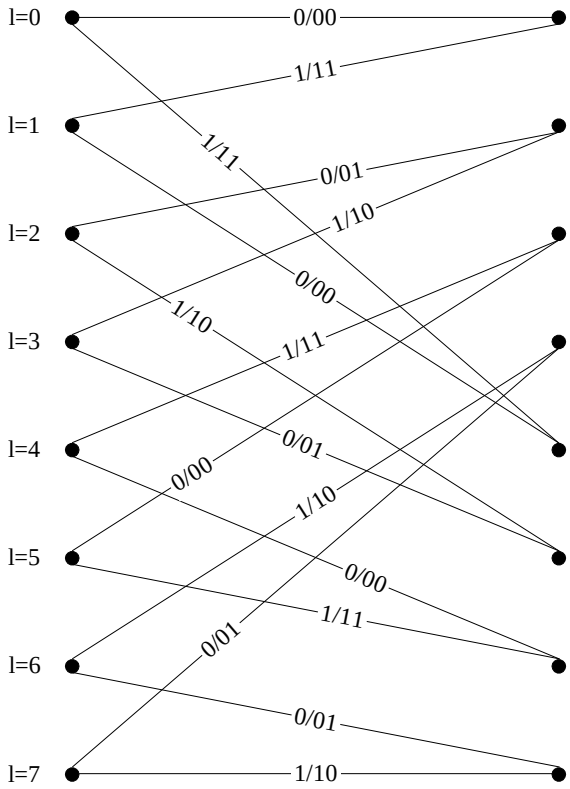

Figure 3.3: Trellis Section of the Optimal 8-State, Rate $R_c = 1/2$ RSC Code, Defined by *Eq.* (3.12)

3.3.2 LINEAR BLOCK CODE DECODERS

Since the classic algebraic algorithms employed in the decoding of linear block codes are well known, these algorithms are not repeated in this dissertation. However, the interested reader is referred [47,93,94] for descriptions of the classic syndrome and ML decoding techniques used for most classes of binary linear block codes. The *Berlekamp-Massey* syndrome decoding algorithm [74, 75], which is the classic hard decision decoding algorithm, employed for both BCH and RS block codes, is addressed in *Appendix B*. Since it falls beyond the scope of this study, the classic *Berlekamp-Massey* algorithm is not described in much detail. However, several valuable references that focus on variations of this decoding algorithm are cited for the interested reader.

Chapter 4 focusses on the trellis decoding of linear block codes by means of a fixed window (or block-wise) VA. This chapter not only describes this decoding algorithm in detail, but also investigates the construction of BCJR block code trellises.

3.3.3 DE-INTERLEAVERS

De-interleavers are in actual fact also interleavers. Their function, however, is to undo the temporal ordering of the symbols, created by the associated interleaver. The following subsection presents a short mathematical description of de-interleavers, building on *Section* 3.2.3.1's discussion of interleavers.

3.3.3.1 MATHEMATICAL DESCRIPTION OF DE-INTERLEAVERS

If the m^{th} length-N sequence of interleaved symbols used as input to a de-interleaver, π^{-1}, associated with an interleaver π, is denoted by $\overline{\varrho}_m^{in} = \{\varrho_{m,0}^{in}, \varrho_{m,1}^{in}, ..., \varrho_{m,(N-2)}^{in}, \varrho_{m,(N-1)}^{in}\}$, the de-interleaving of $\overline{\varrho}_m^{in}$ by π^{-1} can be described as follows [87]:

$$\pi^{-1}\left(\overline{\varrho}_m^{in}\right) = \overline{\varrho}_m^{out} = \{\varrho_{m,0}^{out}, \varrho_{m,1}^{out}, ..., \varrho_{m,(N-2)}^{out}, \varrho_{m,(N-1)}^{out}\} \tag{3.48}$$

where $\overline{\varrho}_m^{out}$ represents the N-symbol/sample output of the de-interleaver. The de-interleaver output can also be written as [87]:

$$\begin{aligned}\overline{\varrho}_m^{out} &= \{\varrho_{m,0}^{out}, \varrho_{m,1}^{out}, ..., \varrho_{m,(N-2)}^{out}, \varrho_{m,(N-1)}^{out}\} \\ &= \{\varrho_{m,\Pi^{-1}(0)}^{in}, \varrho_{m,\Pi^{-1}(1)}^{in}, ..., \varrho_{m,\Pi^{-1}(N-2)}^{in}, \varrho_{m,\Pi^{-1}(N-1)}^{in}\}\end{aligned} \tag{3.49}$$

where $\Pi^{-1}(i)$, with i any integer value, is the de-interleaver mapping function that describes the mapping of the de-interleaver output time indices to de-interleaver input time indices. The mapping function, $\Pi^{-1}(i)$, is defined such that it will undo the temporal shuffling caused by the mapping function $\Pi(i)$ of the interleaver π. For example, the mapping function for the de-interleaver associated with the example period $N = 3$ convolutional interleaver, defined by the mapping function of *Eq.* (3.39), is as follows:

$$\Pi^{-1}(i) = \begin{cases} i & \text{if } i \bmod 3 = 0 \\ i+3 & \text{if } i \bmod 3 = 1 \\ i+6 & \text{if } i \bmod 3 = 2 \end{cases} \tag{3.50}$$

From this mapping function, the de-interleaver's fundamental permutation follows readily:

$$\eth\left(\Pi^{-1}\right) = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 4 & 8 \end{pmatrix} \tag{3.51}$$

Following a similar approach as with interleavers, the stream of de-interleaver input sequences can be presented by the N-dimensional vector sequence $\overline{\varrho}(D)$, given by:

$$\overline{\varrho}^{in}(D) = \sum_m \overline{\varrho}_m^{in} D^m \tag{3.52}$$

where D represents one de-interleaver delay period of N symbols. Defining the generator matrix of the de-interleaver $G_{\pi^{-1}}(D)$, the de-interleaver output stream can be described as follows:

$$\overline{\varrho}^{out}(D) = \overline{\varrho}^{in}(D) \cdot G_{\pi^{-1}}(D) \tag{3.53}$$

Obviously the de-interleaver generator matrix is an $N \times N$ non-singular matrix with the same restrictions as $G_\pi(D)$. Assuming perfect channel conditions, employing interleaver π's output (see *Eq.* (3.43)) as input for the de-interleaver π^{-1}, *Eq.* (3.53) can be rewritten as follows:

$$\overline{\varrho}^{out}(D) = \overline{\varrho}^{in}(D) \cdot G_{\pi^{-1}}(D) = \left(\overline{\mu}^{in}(D) \cdot G_\pi(D)\right) \cdot G_{\pi^{-1}}(D) \tag{3.54}$$

Since the de-interleaver π^{-1} has to undo the shuffling of the interleaver π, i.e. $\overline{\varrho}^{out}(D) = \overline{\mu}^{in}(D)$, the de-interleaver's generator matrix can be determined as follows [87]:

$$G_{\pi^{-1}}(D) = G_\pi^{-1}(D) \tag{3.55}$$

For example, the generator matrix associated with the de-interleaver mapping given by *Eq.* (3.50) must be the inverse of the matrix given by *Eq.* (3.44):

$$G_{\pi^{-1}}(D) = G_{\pi}^{-1}(D) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & D^{-1} & 0 \\ 0 & 0 & D^{-2} \end{bmatrix} \tag{3.56}$$

From this generator matrix it is clear that the example de-interleaver is non-causal, as will always be the case for de-interleavers associated with causal interleavers [87].

3.3.4 CODE DE-PUNCTURERS

Code de-puncturing, also known as *insertion*, is the process whereby the bits punctured from a stream of encoded bits prior to modulation, are reinserted into the received and demodulated stream before being decoded. Since de-puncturing fills all the "holes" in the stream of encoded bits used as input by the decoder, decoding can be accomplished by the standard decoding structure associated with the transmitter's encoder, eliminating the need for an altered decoder, which has been adapted to accommodate a punctured stream of bits as input [105].

An important question now arises: Were the punctured code bits ones or zeros? Since the receiver has no way of knowing the answer to this question, the best solution is to declare *erasures* in the deleted bit positions. Erasures can be though of as "I am not sure" values, the size of which depends on the code bit alphabet employed by the encoder/modulator and demodulator/decoder, as well as the a-priori probabilities for the code bit ones and zeros. The erasure value $\Gamma_{m,i,a}$ for the a^{th} delete code bit in the i^{th} symbol of the m^{th} n-symbol $GF\left(2^{\xi}\right)$ channel coder output can be calculated as follows for slow Rayleigh flat fading channel conditions with AWGN noise effects: Assuming the receiver has perfect knowledge of the instantaneous channel phase and no puncturing was performed, coherent demodulation renders the following demodulator output for the a^{th} bit in the i^{th} symbol of the m^{th} set of received code word symbols:

$$y_{m,i,a} = \overline{\alpha}_{m,i,a}.c_{m,i,a} + \eta_{m,i,a} \tag{3.57}$$

where $\overline{\alpha}_{m,i,a}$ and $\eta_{m,i,a}$ represents the average fading amplitude (see *Section* 5.2.3 and *Section* 5.3.3) and AWGN demodulator output components for this code bit, respectively. With the demodulator output defined by *Eq.* (3.57), it can be shown that the conditional PDF $\rho\left(y_{m,i,a}|c_{m,i,a}\right)$ is given by [60]:

$$\rho\left(y_{m,a,i}|c_{m,a,i}\right) = \frac{1}{\sqrt{2\pi}\sigma_{\eta_{m,i,a}}} \exp\left[-\frac{\left(y_{m,i,a} - \overline{\alpha}_{m,i,a}.c_{m,a,i}\right)^2}{2\sigma_{\eta_{m,i,a}}^2}\right] \tag{3.58}$$

where $\sigma_{\eta_{m,i,a}}^2$ is the variance in the AWGN component. Furthermore, assume the code bit can take on values from the amplitude alphabet $\{C_{zero}, C_{one}\}$. Calculation of the appropriate erasure value $\Gamma_{m,i,a}$ involves finding the intersection between the PDFs for the demodulator output, given $c_{m,a,i} = C_{zero}$ or $c_{m,a,i} = C_{one}$ was transmitted [81]:

$$\begin{aligned} &\text{Prob.}\left(c_{m,a,i} = C_{zero}\right).\rho\left(y_{m,a,i}|c_{m,a,i} = C_{zero}\right)\big|_{y_{m,a,i}=\Gamma_{m,i,a}} = \\ &\text{Prob.}\left(c_{m,a,i} = C_{one}\right).\rho\left(y_{m,a,i}|c_{m,a,i} = C_{one}\right)\big|_{y_{m,a,i}=\Gamma_{m,i,a}} \end{aligned} \tag{3.59}$$

where $\text{Prob.}\left(c_{m,a,i} = C_{one}\right)$ and $\text{Prob.}\left(c_{m,a,i} = C_{zero}\right)$ represent the a-priori probabilities of transmitting ones and zeros, respectively. Solving *Eq.* (3.59) yields the following expression for the

optimal erasure value:

$$\Gamma_{m,i,a} = \frac{\log\left(\frac{\text{Prob.}(c_{m,i,a}=C_{one})}{\text{Prob.}(c_{m,i,a}=C_{zero})}\right).2\pi\sigma^2_{\eta_{m,i,a}} - (\overline{\alpha}_{m,i,a}.C_{one})^2 + (\overline{\alpha}_{m,i,a}.C_{zero})^2}{(2.\overline{\alpha}_{m,i,a}.C_{zero} - 2.\overline{\alpha}_{m,i,a}.C_{one})} \tag{3.60}$$

For example, assuming an antipodal code bit representation from the alphabet $\{-1, 1\}$ for equiprobable code bits, the de-puncturer will replace the deleted code bits with $\Gamma_{m,i,a} = 0$.

Although de-puncturing has a definite influence on the performance of ML and MAP decoders, the gains obtained by employing punctured low rate codes surpass the gains of higher rate codes with equivalent overall code rates [105]. Furthermore, the combination of code puncturing and de-puncturing allows the use of a single code over a wide variate of transmission rates, without requiring alteration of the encoder or decoder structures. As such, code puncturing is frequently encountered in coded communication systems that support rate adaptation.

3.3.5 CHANNEL STATE INFORMATION ESTIMATORS

Although the accurate estimation of mobile communication channel parameters has always been an area of particular appeal to communication engineers, classic channel code designers refrained from using CSI in their decoder algorithms. This was due to the fact that the performance improvements obtained using side information in classic ML decoding did not necessarily justify the required increased system complexity [56]. However, the introduction of TCs in 1993 sparked a renewed interest into this research field, since these codes, like most modern concatenated channel codes, require the use of reliable CSI during iterative decoding [87, 108].

Explained in terms of *Chapter* 2's mathematical framework for mobile communication channels, CSI estimator structures, integrated into current and future wireless communication systems, must accomplish one or more of the following tasks:

1. Estimate the AWGN present at the output of the receiver/demodulator in order to predict the current operational E_b/N_0. Accurate knowledge of the SNR per bit [47] is crucial during the iterative decoding of PCCs, SCCs, HCCs, product codes and LDPCs. This parameter forms an integral part of the *Log-Likelihood Ratio* (LLR) calculations performed by each of the *Soft-Input Soft-Output* (SISO) decoding modules (for example, MAP and SOVA decoders) incorporated into these codes' iterative decoder structures [28, 30–33, 87, 108].

2. Determine the power delay profiles (or impulse responses) of multipath fading channels [109]. For an L-path channel, this entails estimating the average path powers $\overline{\beta}_i$ and path delays τ_i, for $i = 1, 2, ..., L$. For wideband DS/SSMA communication systems, full exploitation of the potential diversity gains achievable by using MRC during RAKE reception, is only possible if these parameters are perfectly known at all time instances [110]. Power control algorithms in DS/SSMA systems also require such knowledge in order to avoid near-far problems in CDMA environments [111].

3. Approximate the fading amplitude and phase for each propagation path in an L-path multipath fading channel, i.e. find $\alpha_i(t)$ and $\phi_i(t)$, for $i = 1, 2, ..., L$. Tracking of the phase changes introduced by the channel (as well as other system components, such as filters), are essential not only for coherent demodulation [86], but also for bit, frame and sequence (in DS/SSMA systems) synchronisation. On the other hand, obtaining real-time estimates of the fading amplitudes are not crucial to the signal detection process. However, these estimates are an integral part of the metric calculations of any channel coded system employing CSI-enhanced decoding. Although perfect tracking of these parameters for each path in a multipath fading channel is desired, implementation

thereof might require excessive system complexity. Therefore, most hardware implementations of RAKE receivers (see *Section* 5.3.2) usually only attempt to track the fading amplitudes and phases of the dominant propagation path, partially forfeiting information carried by the precursor and post-cursor [47, 112] paths.

4. Efficient multi-user detection in CDMA systems is a crucial component of most multi-user cancellation techniques, which are geared at minimising the effects of the MUI present in the channel [43, 44, 113].

Over the last few decades, numerous channel parameter tracking methods and algorithms have been proposed. However, this field is still in its infancy. A universal channel estimation technique, capable of handling all modulation schemes and mobile channel characteristics, still eludes the communication engineering society. Current schemes are mostly application specific, integrated into the modulation technique, MA scheme, channel equalisation subsystem or channel coding/decoding algorithms employed by modern communication systems. That being said, it is still possible to organise the myriad of proposed schemes into two main categories:

1. **Blind Channel Estimation:** With this approach, no additional information, which might assist with CSI estimation, is embedded into data transmissions. At the receiving end, channel parameters must be extracted directly from the modulated information signals. Obviously, schemes falling in this category are highly complex, but also much sought after, since no transmission bandwidth is relinquished for CSI estimation purposes. Many classic carrier, sequence, bit and frame synchronisation loops fall in this category, for example *Costas* loops [47], decision directed early-late code locked loops [43, 47], etc. Multipath fading channel impulse response (power delay profile) and AWGN power level estimations via first, second and higher order statistics (calculation of mean values, variances, auto-correlations, cross-correlations, etc.) can also be grouped into this category. One might also consider free-running equalisers (after successfully training) to be blind channel estimators. Unfortunately, very few fading amplitude estimation schemes exist that can be considered to be purely blind. However, for constant envelope transmitter output signals, blind estimation of the fading amplitude is easily accomplished.

2. **Pilot Assisted Channel Estimation:** This approach relies on the use of dedicated pilot information (such as pilot tones, pilot symbols, etc.) from which channel parameters can effortlessly be determined. This category can be further subdivided into:

    1. In-band pilot signalling, where pilot signalling occupies the same bandwidth as the information being transmitted. For example, with *Pilot Symbol Assisted Modulation* (PSAM), non-information carrying symbols are injected directly into the transmitted data stream, effectively sacrificing a percentage of the transmission bandwidth. At the receiving end, *Kalman* filters [114, 115] can be used to interpolate between the amplitude changes observed in consecutive pilot symbols, thereby estimating the fading amplitude of the mobile channel. Another example of in-band signalling is the use of equaliser training sequences, present in every normal burst of GSM.
    2. Out-of-band pilot signalling, where dedicated bandwidth is assigned for pilot signalling purposes. A system that uses a pilot channel to carry a pilot tone for carrier synchronisation purposes, is a good example of out-of-band pilot signalling.

Although an in depth investigation into channel estimation schemes falls outside the scope of this study, the list of journal articles and conference papers below attempts to spark the curiosity of the interest reader:

- PSAM techniques for the estimation of flat fading channel parameters and statistics are discussed and analysed in [116–118].

- *Maximum Likelihood Sequence Estimation* (MLSE) algorithms (such as the VA) and per-survivor processing for channel estimation purposes are covered in [119–123].
- In [109, 124–126] the estimation of multipath fading channel power delay profiles (impulse responses), as well as other channel statistics are addressed.
- Several joint data detection and channel estimation approaches are considered in [122, 123, 125] for flat and frequency selective fading channels.
- Blind equalisation and channel estimation techniques, including second order cyclostationary statistical methods, the use of chaotic coded signals, least-squares approaches, linear prediction methods, periodic modulation precoders, *Soft Output Viterbi Equaliser* (SOVE) algorithms and tricepstrum-based algorithms are presented in [127–136].
- Various multipath fading channel estimation techniques, targeted at DS/SSMA systems, are given in [111, 137, 138]. In [139] the influence of channel state estimation on the performance of a coherent DS/SSMA system is investigated.
- A technique for multi-user detection through adaptive channel estimation is described in [140].

Chapter 6 presents a great number of simulated BER performance curves for convolutional and linear block codes, employing soft decision VA decoding with perfect fading amplitude CSI. This information was directly extracted (see *Section* 2.6.2.5) from the novel complex flat fading and multipath fading channel simulator structures, presented in *Section* 2.6.2.3 and *Section* 2.6.3.2, respectively.

3.4 CONCLUDING REMARKS

The encoder and decoder building blocks encountered in classic block and convolutional coding schemes were considered in this chapter. This included discussions on binary convolutional codes, binary and non-binary linear block codes, interleaver and de-interleaver structures, the concept of code puncturers and de-puncturers, and last, but not least, CSI estimators. The following unique, albeit insignificant contributions were made in this chapter:

1. A unified generator matrix approach is employed to describe convolutional codes, linear block codes, interleavers and de-interleavers. Although this is commonplace for linear block codes, the same can not be said for binary convolutional codes, interleavers or de-interleavers.
2. In *Section* 3.3.4, which focuses on the issue of code de-puncturing, a simple formula is presented that calculates an optimal erasure value for non-equiprobable code bits, transmitted through a slow Rayleigh flat fading channel with AWGN effects.