Chapter 1

Introduction

This chapter provides the framework on which the rest of the dissertation is based.

To start off, a historical overview is presented stretching back to the dawn of modern telecommunication over half a century ago. Following this historical account, a clear breakdown of the topics examined and contributions made is provided. This will provide the reader with an overview of the work following the introduction.

In the next section, the goals and objectives of the dissertation are outlined. They are again evaluated in the summary at the end of the dissertation, and should be read together with this introductory chapter.

1.1 Historical Overview

Since the epochal work by Claude E. Shannon in 1948 [23], various different channel encoder and channel decoding techniques have been developed for the transmission of data over noisy communication channels. The aim of all these techniques are the same: Transmit as much data as possible given a certain time frame and channel characteristic with the least amount of errors. This simple statement caused an uproar in the communications and mathematical fields when Shannon derived a bound specifying the maximum capacity that can be achieved at various energy to noise ratios for a given channel. Since 1948 this bound is commonly known as the *Shannon Bound* [23]. It has become the dream of many a designer and researcher to cross this theoretical bound discovered by Shannon. To date, the bound has not been reached nor exceeded, although the latest techniques come to within dB's of this elusive goal.

The various coding techniques are abound as sand in the Sahara desert. All channel coding techniques can be divided into two main groups, namely the class of convolutional codes and the class of block codes. Many hybrid systems have also evolved over the years, but in principle, the two above mentioned groups are the two main fields on which

research is currently done.

The enormous difference between the properties of these two groups of codes has always lead to either block codes or convolutional codes being researched, but efforts involving combinations of these codes were few and far between. This phenomenon mainly sprouted from the completely different decoding procedures used to decode the two groups of codes. For block codes, pure algebraic techniques involving correlative and comparative processes were developed, whereas elegant soft decision techniques were discovered for convolutional codes. As the convolutional soft decision decoding techniques improved with time, the use of block codes in systems became less attractive. This spiraling effect led to a situation were almost all of the effort was being focused on convolutional codes and their decoding techniques.

The decoding techniques for convolutional codes rely on the fact that all convolutional codes can be described by a state-time variant diagram termed a *trellis* diagram. This sparked the idea that if a trellis diagram could be obtained for a block code, the same elegant convolutional decoding techniques could be applied in the decoding process of block codes.

In 1974 Bahl, Cocke, Jelinek and Raviv [2] proposed a possible method for the construction of a block code trellis diagram. Their valuable work was used as a base by J.K. Wolf [16] in 1978 in order to develop a procedure for the construction of block code trellis diagrams. Forney [5] also elaborated on the topic, and introduced the world to *coset trellises*, which are described in the chapters to follow.

Parallel to this discovery, a new type of block code was introduced on the 21st of January 1959. This non-binary block code became known as the Reed-Solomon[20] code, named after its discoverers, Irvine Reed and Gus Solomon. The Reed-Solomon code has become the most widely used block code and has the best error correcting capability of all block codes. Many different techniques, apart from the algebraic ones, have been developed to decode Reed-Solomon codes (the Berlekamp Massey [21] technique to name just one). Nevertheless, the reasons why this group of codes has never enjoyed the advantages of convolutional decoding techniques can firstly be attributed to the enormous size of the codes and secondly to the fact that it falls into the category of a block code. This dissertation will investigate methods for soft decision maximum likelihood decoding of Reed-Solomon codes. A quote from a paper by Cooper [21] is found to be very relevant

here: "*The Holy Grail of Reed-Solomon decoding research is the soft decision maximum likelihood decoder. There is however still a great deal of work to be done before the Grail is found and the knights can go home.*"

Now, 50 years later, more and more effort is channeled into the development of novel channel encoding techniques employing both block codes and convolutional codes. In this manner, it is aimed to combine the best of two worlds in order to get closer to the ultimate goal in communication theory - the Shannon Bound.

1.2 Goals and Objectives

The dissertation is mainly concerned with various decoding techniques of block codes. The main objective is to decode block codes with the same decoders that are employed in the decoding process of convolutional codes, namely the Viterbi and MAP algorithms.

From this main goal several secondary goals emanate which are listed below. Each secondary goal is considered to be as important as the original goal.

- This dissertation aims at providing viable trellis construction techniques for block codes, their success being evaluated and judged in terms of complexity.
- Soft decision maximum likelihood techniques, traditionally reserved for the decoding of convolutional codes, are applied to the decoding process of block codes.
- Comparisons between the Viterbi decoding techniques and the algebraic techniques are made. The comparisons are done for both hard decision and soft decision decoding.
- The trellis construction techniques are then extended to include the construction of trellises for the Reed-Solomon family of codes.
- A novel trellis construction technique for Reed-Solomon codes is presented which reduces the amount of calculations required during the construction of such a trellis.

- Decoding of the Reed-Solomon codes is attempted employing the Viterbi algorithm.

- The dissertation shows that maximum likelihood decoding can be achieved with the techniques mentioned above, and that it compares favorably with the traditional algebraic decoding results.

- An algorithm is investigated which decreases trellis complexity, making decoding of larger block codes possible with the Viterbi algorithm. Simulation results are presented to verify the decoding performance using the lower complexity trellises.

1.3 Outline of Dissertation

In this chapter, the *Introduction*, a framework is provided on which the entire dissertation is based.

The second chapter, *Foundations of Channel Coding*, deals with important coding and information theory concepts. The emphasis in this chapter is placed on coding theory, and all the relevant topics are discussed. The most important aspect of the coding theory as far as this dissertation is concerned, is the discussion of block codes and their properties. This discussion will provide the foundation on which all other chapters are based. Throughout the remainder of the work, reference will be made to the second chapter of the dissertation.

Decoding of Block Codes is the third chapter of the dissertation, and employs the technique discussed in the previous chapter to realise a method for decoding block codes. Here techniques normally reserved for the decoding of convolutional codes are adapted and used for the decoding of block codes. The main focus in this chapter is the creation of trellis diagrams for block codes. This allows the aforementioned decoding processes to be used for block codes. Not all the known trellis construction techniques are discussed in this chapter, but reference is made to the appendixes at the end of the dissertation for more information.

The chapter, *Performance Issues of Various Block Codes*, investigates the performance of a selected group of block codes, namely the Hamming, Reed-Muller and BCH families

of codes. Error rate curves of the block codes obtained by using the Viterbi algorithm are compared to curves obtained by traditional algebraic decoding techniques. These curves are then also compared to the theoretical curves obtained by calculation. It is shown that maximum likelihood decoding is achieved by employing the convolutional decoding techniques on block codes.

Trellis Construction for Reed-Solomon Codes is discussed in chapter 5. The importance of this group of error correcting block codes has already been pointed out. Very little information is available on this topic in literature, and therefore a great amount of detail is presented in this chapter. Again, the first step for Viterbi decoding of Reed-Solomon codes is the construction of a trellis diagram for the code. As Reed-Solomon codes are huge compared to any other type of code, their trellis diagrams are too large to illustrate schematically. For this reason a small code was chosen for illustration purposes. A topological analysis is done on the constructed trellis, and several novel findings are made, resulting in a novel "topological" trellis construction technique for Reed-Solomon codes. The performance of this family of codes, when used together with convolutional decoders, is examined. Error rate curves are plotted in order to demonstrate that decoding of Reed-Solomon codes employing either the Viterbi or MAP algorithms is in fact possible. This ultimately means that Reed-Solomon codes can be maximum likelihood soft decision decoded by the same algorithms normally used for convolutional codes.

Chapter 6 presents an introduction to trellis complexity, which is accompanied by a technique for the successful reduction of trellis complexity. As mentioned earlier, the only limiting factor in the trellis decoding process is the actual complexity of the trellis. The larger the trellis, the more complex the decoding, and this will ultimately make the decoding of large block codes impractical. A simple example is used to show that this proposed technique is in fact able to reduce the trellis complexity by a significant factor. The effect of this technique is that larger codes can now be decoded with the same amount of effort required to decode smaller codes, since the trellis size of the larger codes can be reduced significantly. This counteracts the apparent drawback of block code decoding with the Viterbi Algorithm. Simulation results are presented which show that there is no degradation in error correcting performance when such a reduced trellis is used in the decoding process.

A *Summary* is provided in chapter 7. As mentioned earlier, this chapter should be read

together with the current one in order to determine which of the goals and objectives have been achieved. An objective stance is taken and the goals are evaluated in the last chapter of the dissertation.

Apart from the chapters listed above, there are a total of eight Appendixes which provide more information regarding several topics under discussion to the interested reader.

1.4 Major Contributions

The major contributions made in this work are as follows:

- Trellis complexity resolution.
- Novel trellis construction techniques based on Wolf's method.
- Simulation results for verification of techniques.

Chapter 2

Foundations of Channel Coding

2.1 Introduction

This chapter is dedicated to describing the general principles of channel coding and error correcting codes. The two main groups of error correcting codes that will be discussed here, are block codes [3], [4], [8] and convolutional codes. In order to describe exactly how a trellis for a block code is constructed in the following chapter, it is essential that several definitions and descriptions be given in this chapter. For this reason, this chapter is divided into four parts. The first part gives a general description of the fundamentals of block coding. This is then followed by a description of convolutional codes. The third part illustrates several decoding principles, along with a detailed derivation of metrics. At the end of this section, a short description of general methods for block code decoding will be given. The last section will detail the Viterbi algorithm [14], as this is, besides the MAP algorithm, still the most important algorithm available for decoding block code trellises.

2.2 Block Codes

Besides providing a mathematical framework for block codes[3], [4], [8], [15], this section mainly contains a description of block codes. Several coding terms such as Hamming-weight, Hamming-distance, minimal-distance and error correcting capability, which are used freely throughout this work and other standard works on this topic, will be described and illustrated. The prior knowledge of material contained in this chapter is of utmost importance before attempting to master any additional information and methods contained within this work.

2.2.1 Linear Block Code, Generator Matrix and Parity Check Matrix

The vectors of the n-dimensional vector space $GF(q)^n$ contain n components, which are taken from the symbol alphabet $GF(q) = \{0,1,...,q-1\}$. If any arbitrary subset of this vector space contains q^k vectors, then this set is termed a (n, k)-block code C with code rate $R = k/n$ and its elements are the codewords $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$. If the addition of two codewords and the multiplication of one codeword with an element of $GF(q)^n$ gives another codeword, the code will be a subset of $GF(q)^n$ termed a *linear block code*. From the above reasoning, it follows that the zero vector **0** has to be contained within every linear block code.

If the possible 2^k information vectors $\boldsymbol{i} = (i_0, i_1, \ldots, i_k)$ are each assigned to a codeword C in a specific way, a code book is formed. The specific manner of assigning these vectors is of utmost importance, as this is what differentiates one code in the space from another one. A linear assignment proves very effective with linear block codes, as the code can then be described by a multiplication of the information vector **i** and a $(k \times n)$-matrix G:

$$\boldsymbol{c} = \boldsymbol{i} \cdot \boldsymbol{G} \tag{2.1}$$

The matrix **G** with elements from $GF(q)$ has a rank k and is termed the *generator matrix*. The row vectors $\mathbf{g}^i$ form a possible base for the subset of $GF(q)^n$ described by C, in that every codeword of C is a linear combination of the row vectors $\mathbf{g}^i$ of **G**, with $i = 0,1,...,k$. The vectors $\mathbf{g}^i$ can be described as either a column vector or a row vector, depending on whether a subscript or superscript notation is used, i.e.

$$\textit{Column vectors} \quad g_j = \begin{pmatrix} g_{1j} \\ g_{2j} \\ \vdots \\ g_{kj} \end{pmatrix}, \qquad j = 1,\ldots,n \tag{2.2}$$

$$\textit{Row vectors} \quad g^i = (g_{i1}, g_{i2}, \ldots, g_{in}), \quad i = 1,\ldots,k \tag{2.3}$$

with

$$\boldsymbol{G} = (g_{ij}) = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{km} \end{pmatrix} = (g_1, g_2, \ldots, g_n) = \begin{pmatrix} g^1 \\ g^2 \\ \vdots \\ g^k \end{pmatrix}. \tag{2.4}$$

The set of all vectors, which are orthogonal to the vectors of C, the so called orthogonal complement $C_\perp$ of C, is called the dual code of C. The dimensions of this code is n - k. If any n - k base vectors of $C_\perp$ are randomly chosen for the rows $\boldsymbol{h}^i$ of the $((n - k) \times n)$-matrix **H**, then the following holds

$$\boldsymbol{c} \cdot \boldsymbol{H}^T = 0, \tag{2.5}$$

where 0 is the zero vector of length n - k. The matrix **H**, which already identifies the linear block code C without the use of the accompanying generator matrix **G**, is termed the *parity check* matrix.

Since the rows $\boldsymbol{g}^i$ of the generator matrix **G** are possible codewords, it follows from equation (2.5), that

$$\boldsymbol{G} \cdot \boldsymbol{H}^T = 0, \tag{2.6}$$

where **0** is the $(k \times (n - k))$-zero vector.

2.2.2 Equivalent Codes and Systematic Codes

If a (n,k) block code C is defined by its generator matrix **G**, an equivalent code with the same parameters can be constructed if the following operations are performed[8]:

- Elementary row operations:
  - Exchanging two rows
  - Multiplication of a row with a field element $\alpha \in (GF(q) \setminus \{0\})$
  - Replacing a row by the addition of that same row and a multiple of any other row
- Exchanging any column vectors

If a chosen number of operations are performed in succession, then the resulting manipulation can be described by a matrix multiplication $\mathbf{T}_G \cdot \mathbf{G}$, where $\mathbf{T}_G$ is a regular $(k \times k)$ matrix with elements taken from $GF(q)$. With this type of manipulation, it is possible, without changing the original code C (the arrangement of the code vectors $\mathbf{c}^i$ do however change), to rearrange any generator matrix into a form where k of the n columns are represented by unit vectors of length k. By exchanging the appropriate k rows, the generator matrix can be transformed to:

$$\boldsymbol{G}' = (\boldsymbol{I}_k | \boldsymbol{A}). \tag{2.7}$$

In the above equation $\mathbf{I}_k$ is the $(k \times k)$ unit matrix and $\mathbf{A}$ is a $(k \times (n - k))$ matrix. A code with a generator matrix in this form is termed a systematic code. The first k code symbols correspond with the information vector $\mathbf{i}$, since

$$\boldsymbol{c} = \boldsymbol{i} \cdot \boldsymbol{G}' = (\boldsymbol{i} | \boldsymbol{i} \cdot \boldsymbol{A}) = (i_0, i_1, \ldots, i_{k-1}, c_k, c_{k+1}, \ldots, c_{n-1}). \tag{2.8}$$

where the $\{c_j\}$, $j = k,\ldots,n - k$ denote the parity bits of the codeword C.

Although not explicitly proven here, it is shown that every block code can be transformed into a corresponding systematic code.

If the generator matrix $\mathbf{G}$ is transformed into the systematic form as per equation (2.7), then the corresponding parity check matrix $\mathbf{H}$ can be found accordingly:

$$\boldsymbol{H}' = (-\boldsymbol{A}^T | \boldsymbol{I}_{n-k}), \tag{2.9}$$

where $\mathbf{I}_{n-k}$ indicates the $((n - k) \times (n - k))$ unit matrix. This is fairly simple to prove, since

$$\boldsymbol{G}' \cdot \boldsymbol{H}'^T = (\boldsymbol{I}_k | \boldsymbol{A}) \cdot \begin{pmatrix} -\boldsymbol{A} \\ \boldsymbol{I}_{n-k} \end{pmatrix} = -\boldsymbol{A} + \boldsymbol{A} = 0 \tag{2.10}$$

as per equation (2.6).

Just like the row and column operations performed on the generator matrix **G** do not affect the code, so does the performing of row and column operations on the parity check matrix not affect the code at all. The entire transformation process of the parity check matrix **H** can be described, as before, by a matrix multiplication $\mathbf{T}_H \cdot \mathbf{H}$, where $\mathbf{T}_H$ is a regular $((n - k) \times (n-k))$ matrix with elements taken from $GF(q)$. Similar to the generator matrix, the exchange of columns again provides an equivalent (but not the same) code.

Often an equivalent form of the original code C may facilitate a far simpler decoding procedure. It is often also desirable (for several reasons) to describe a block code C in its systematic form C'. Simple decoding procedures have also been developed for systematic error correcting block codes.

2.2.3 Cyclic Codes - Generator and Parity Check Polynomials

When any shifted version of a linear block code C produces another valid codeword in C, then this code is labeled *cyclic*. As shown in a few standard works, any cyclic code can be described by a generator polynomial [8] in the form:

$$g(x) = g_0 + g_1 \cdot x + g_2 \cdot x^2 + \ldots + g_{n-k} \cdot x^{n-k} \tag{2.11}$$

In relation to this, the coefficients of the code polynomial

$$c(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + \ldots + c_{n-1} \cdot x^{n-1} \tag{2.12}$$

are taken from the components of the code vector $c = (c_0, c_1, \ldots, c_{n-1})$.

A very simple relationship exists between the code vector $c(x)$ and the information vector $i(x)$, namely that one can directly multiply the information polynomial $i(x)$ of degree $\leq k - 1$ with the generator polynomial $g(x)$, and as a result obtain the code polynomial $c(x)$ of degree $\leq n - 1$.

$$c(x) = i(x) \cdot g(x) \tag{2.13}$$

The resulting generator matrix is given below:

$$\boldsymbol{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & & & 0 \\ & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & & \\ & & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & \\ & & & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & \\ 0 & & & & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} \end{pmatrix} \quad (2.14)$$

Similar to equation (2.5), there exists a so called parity check polynomial $h(x)$ of degree k for cyclic codes for which the following holds true:

$$c(x) \cdot h(x) = 0 \mod(x^n - 1) \quad \textit{for all } c(x) \in C. \quad (2.15)$$

This leads to an expression:

$$g(x) \cdot h(x) = x^n - 1. \quad (2.16)$$

For any possible h(x), the parity check matrix could be given as follows:

$$\boldsymbol{H} = \begin{pmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_2 & h_1 & h_0 & & & 0 \\ & h_k & h_{k-1} & h_{k-2} & \cdots & h_2 & h_1 & h_0 & & \\ & & h_k & h_{k-1} & h_{k-2} & \cdots & h_2 & h_1 & h_0 & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & & & & h_k & h_{k-1} & h_{k-2} & \cdots & h_2 & h_1 & h_0 \end{pmatrix}. \quad (2.17)$$

2.2.4 Syndrome

If a codeword is transmitted over a noisy channel, it is possible that some of the symbols of the transmitted code are in error. In other words, an error vector **e** is added to the code vector **c**:

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e} \quad (2.18)$$

If the received vector r is multiplied with the transposed parity check matrix $\mathbf{H}^T$, then the answer is 0 only if $\mathbf{e} = \mathbf{0}$ (error free transmission), or if $\mathbf{e} \in C$, which in term translates $\mathbf{r}$ into another valid codeword. The expression $\mathbf{r} \cdot \mathbf{H}^T$ is called the *syndrome* s, and is not only dependant on the error vector $\mathbf{e}$, but also on the transmitted codeword c, since it can be shown with the help of equation (2.5) and (2.18) that the following holds true:

$$s = r \cdot H^T = (c + e) \cdot H^T = e \cdot H^T \tag{2.19}$$

2.3 Convolutional Codes

Similar to the previous section, this section presents the basic fundamental concepts of convolutional codes [5], [6]. Convolutional codes are the second large group of error correcting codes. Their structure is vastly different from block codes, and the generation of these codes is also done in a completely different way.

2.3.1 Convolutional Encoder

In the previous section it was shown that the n symbols of a block code were just dependant on the k information symbols of the relevant information vector. However, for a convolutional code, an additional parameter is required, namely M. The parameter M is called the *arrangement of memory* of the encoder, or in short, the *memory* of the convolutional encoder. The n code symbols of the convolutional code are thus a function of M previous information vectors.

An example of a convolutional encoder defined over GF(q) with $q = 2$, $R = k/n = 2/3$ and $M = 1$ is shown in **Figure 1**. For each branch i, $k = 2$ information symbols from the symbol alphabet $GF(q = 2)$ are shifted into the shift register. These are then combined with the previous information vector ($M = 1$) to form a codeword $\mathbf{c}_i$. The shift registers are loaded with zeros at the start of the encoding process.

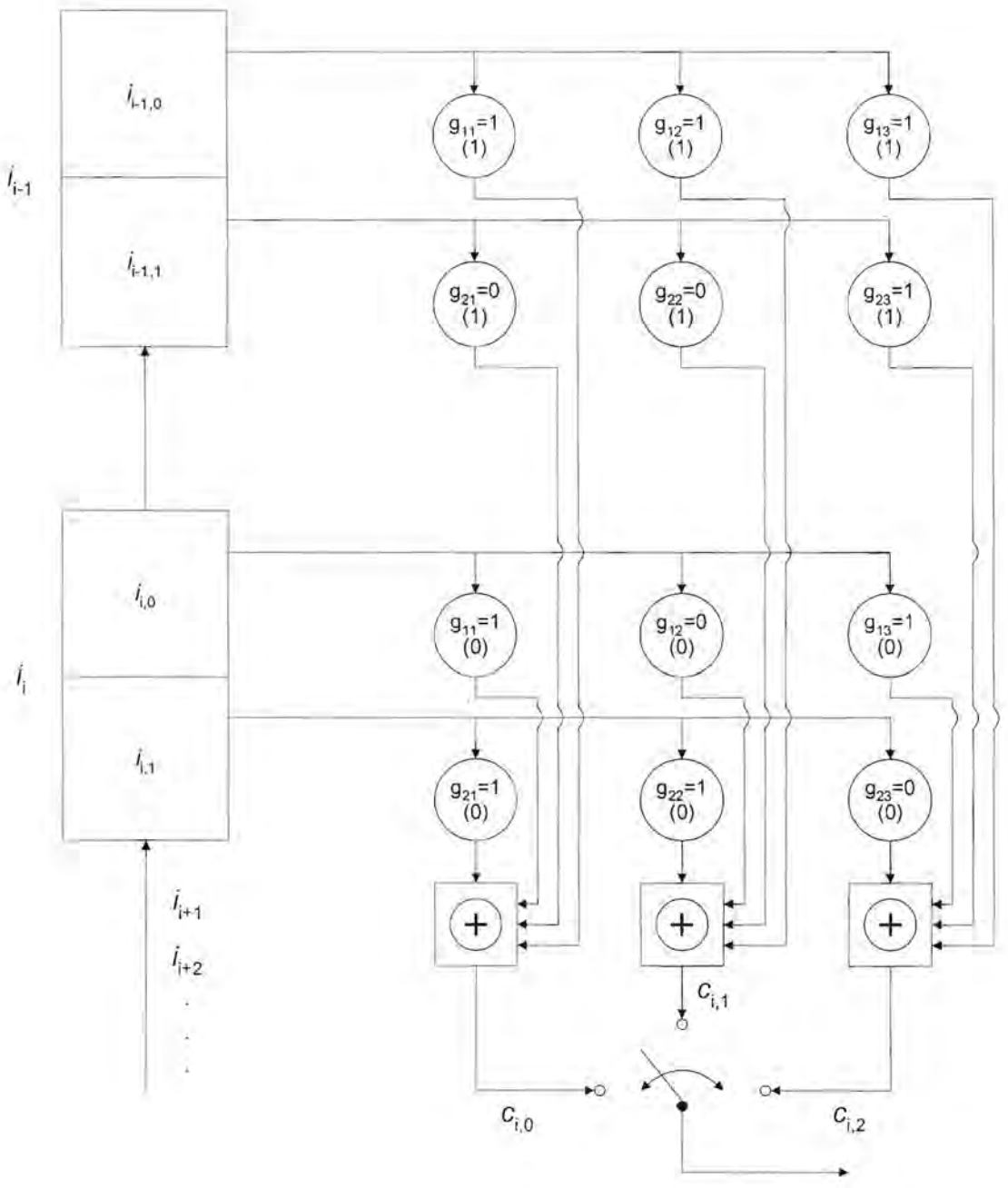

Figure 1 Example of a Convolutional Encoder with R = 2/3 and M = 1

Several methods of describing convolutional codes and convolutional encoders [5], [14] have been developed. For the purposes of this study, only the matrix representation and the representation in trellis form will be described. The above example of a convolutional encoder will be used to illustrate the representation forms.

2.3.2 Matrix Representation of Convolutional Codes

The mapping of the information sequence

$$\boldsymbol{i} = (i_0, i_1, i_2, \ldots, i_l, \ldots) \tag{2.20}$$

into the code sequence

$$\boldsymbol{c} = (c_0, c_1, c_2, \ldots, c_l, \ldots) \tag{2.21}$$

can be described (analog to equation (2.1)) by

$$\boldsymbol{c} = \boldsymbol{i} \cdot \boldsymbol{G} \tag{2.22}$$

where the generator matrix has the following semi-infinite form:

$$\boldsymbol{G} = \begin{pmatrix} G_0 & G_1 & G_2 & \cdots & G_{M-1} & G_M & & 0 \\ & G_0 & G_1 & G_2 & \cdots & G_{M-1} & G_M & \\ & & G_0 & G_1 & G_2 & \cdots & G_{M-1} & G_M \\ 0 & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{2.23}$$

The (k × n) sub-matrixes

$$\boldsymbol{G}_l = (g_{ij}^{(l)}) = \begin{pmatrix} g_{11}^{(l)} & g_{12}^{(l)} & \cdots & g_{1n}^{(l)} \\ g_{21}^{(l)} & g_{22}^{(l)} & \cdots & g_{2n}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1}^{(l)} & g_{k2}^{(l)} & \cdots & g_{kn}^{(l)} \end{pmatrix} \tag{2.24}$$

can be read directly from **Figure 1**. In this example, the matrix is as follows:

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & & & & & & \cdots & & \cdots & \cdots \\ 1 & 1 & 0 & 0 & 0 & 1 & & & \mathbf{0} & & & & \cdots & & \\ & & & 1 & 0 & 1 & 1 & 1 & 1 & & & & & & \\ & \mathbf{0} & & 1 & 1 & 0 & 0 & 0 & 1 & & \mathbf{0} & & \cdots & \cdots & \\ & & & & & & 1 & 0 & 1 & 1 & 1 & 1 & & & \\ & \mathbf{0} & & & \mathbf{0} & & & & & & & & \mathbf{0} & \cdots & \\ & & & & & & 1 & 1 & 0 & 0 & 0 & 1 & & & \\ & & & & & & & & & & \ddots & & \ddots & & \end{pmatrix} \tag{2.25}$$

with

$$\boldsymbol{G}_0 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \tag{2.26}$$

and

$$\boldsymbol{G}_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.27}$$

The generator matrix can also be represented in a polynomial form, as follows:

$$\boldsymbol{G}(D) = (g_{ij}(D)) = \begin{pmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1n}(D) \\ g_{21}(D) & g_{22}(D) & \cdots & g_{2n}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1}(D) & g_{k2}(D) & \cdots & g_{kn}(D) \end{pmatrix}. \tag{2.28}$$

The elements $g_{ij}(D)$ of the above matrix are polynomials of degree $M_{ij} \leq M$

$$g_{ij}(D) = g_{ij}^{(0)} + g_{ij}^{(1)} \cdot D + g_{ij}^{(2)} \cdot D^2 + \ldots + g_{ij}^{(M_{ij}-1)} \cdot D^{M_{ij}-1} + g_{ij}^{(M_{ij})} \cdot D^{M_{ij}} \tag{2.29}$$

with the coefficients $g_{ij}^{(l)}$ of the matrixes $\mathbf{G}_l$ from equations (2.23) and (2.24).

$\mathbf{G}(D)$ can thus be written as

$$\boldsymbol{G}(D) = \boldsymbol{G}_0 + \boldsymbol{G}_1 \cdot \boldsymbol{D} + \boldsymbol{G}_2 \cdot \boldsymbol{D}^2 + \ldots + \boldsymbol{G}_{M-1} \cdot \boldsymbol{D}^{M-1} + \boldsymbol{G}_M \cdot D^M. \tag{2.30}$$

The so called *memory size* of a convolutional encoder is defined as follows:

$$\nu_G = \sum_{i=1}^{k} \max_j M_{ij} \leq M \cdot k \tag{2.31}$$

In this example this would translate to:

$$\boldsymbol{G}(D) = \begin{pmatrix} 1+D & D & 1+D \\ 1 & 1 & D \end{pmatrix} \tag{2.32}$$

and

$$\nu_G = \sum_{i=1}^{2} \max_j M_{ij} = \max(1,1,1) + \max(0,0,1) = 1 + 1 = 2. \tag{2.33}$$

The memory size ν_G exactly determines the amount of registers used in the shift register structure of the convolutional encoder.

2.3.3 Trellis Diagram Representation

If one defines the contents of the registers ν_G containing the previous information symbols, which are used to calculate the current codewords, as states of the convolutional encoder, then a series of nodes develop. These series of nodes allow the representation of the q^{ν_G} possible states, and emanating branches that represent the

q^k possible information vectors at a certain instant in time. This graph of topologically arranged nodes and branches is called a trellis diagram. In short, it can be describes as the time equivalent representation of the state diagram of the convolutional encoder.

For the convolutional encoder presented in **Figure 1** of this section, the corresponding trellis diagram is given in **Figure 2** below.

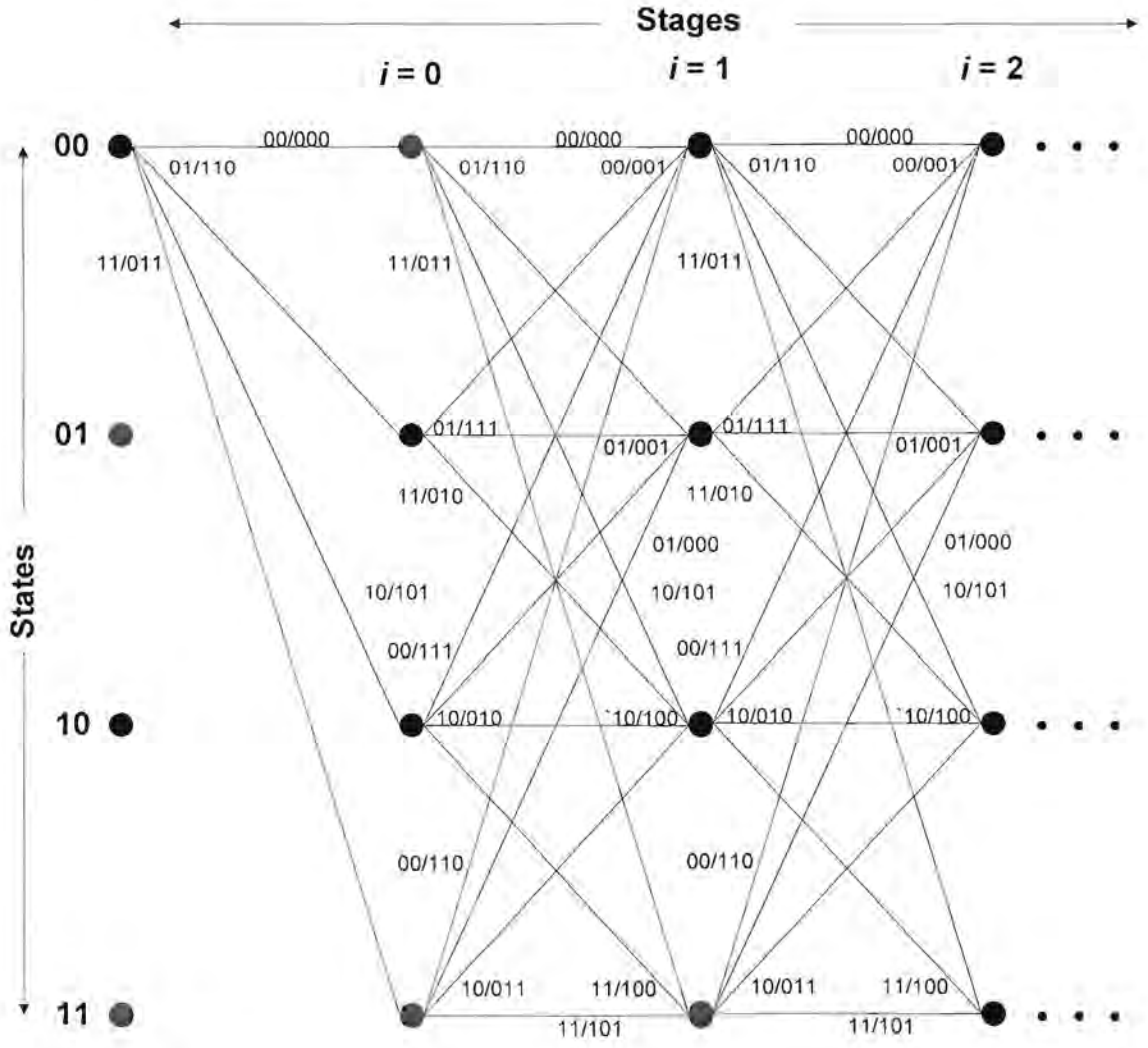

Figure 2 Trellis Diagram of the Convolutional Encoder from Figure 1

The vectors on the left hand side of the figure indicate the state that the specific encoder is in. On the branches, the relevant information vectors along with their respective code vectors are indicated. The notation that is used on the branches is a general

input/output notation, and this notation will be used throughout the dissertation. This means that the information vectors are listed on the left hand side and the resulting output or code vectors are listed on the right hand side.

After M stages, when the information vector has reached the last register block, the trellis is fanned out. This means that all the branch transitions from that stage will be the same as the branch transitions from any stage after that particular stage where fan-out has occurred.

2.4 On the Decoding of Block and Convolutional Codes

In this section, several decoding principles of error correcting codes will be presented. This will be the basis on which the latter chapters on decoding will be based.

2.4.1 General Decoding Principles

In **Figure 3**, a simple structure for a digital transmission system [10] utilizing a channel encoder and a channel decoder is shown. At the encoder, k information symbols (an information vector $\mathbf{i}$) are mapped onto n code symbols (a codeword $\mathbf{c}$). The code sequence is then encoded modulation specifically in the block "*Transmitter*" and transformed into a transmission signal (*modulation*). This signal is then transmitted over the "Additive White Gaussian Noise" channel, where transmission errors are being introduced to the original error free signal. In the "*Receiver*" block, the received signal is demodulated to produce the estimated received sequence $\mathbf{r}$. The decoder establishes an estimated equivalent $\hat{\mathbf{i}}$ of the transmitted information sequence $\mathbf{i}$.

For simplicity reasons, a BPSK-modulation [10] is assumed. Furthermore, no inter-symbol interference is found in the channel. If this is the chase, then the blocks labeled "*Transmitter*", "*AWGN channel*" and "*Receiver*" in **Figure 3**, can be combined into a time-discrete memory-free channel, in which the j^{th} component of the received sequence $\mathbf{r}$ is only dependant on the j^{th} component of the code sequence $\mathbf{c}$.

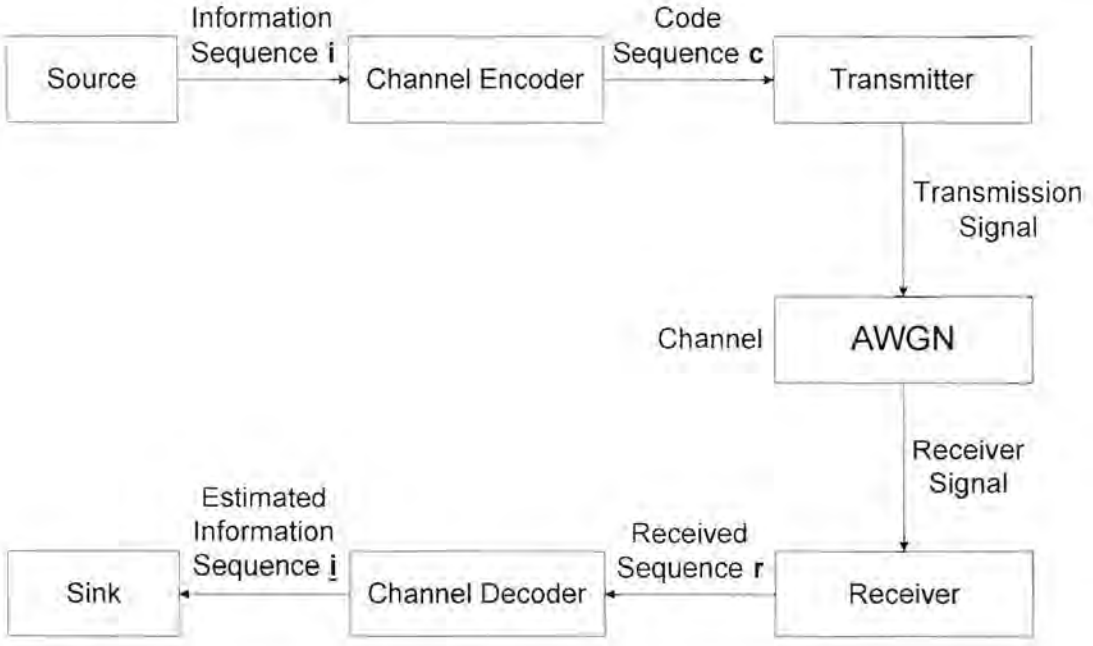

Figure 3 Structure of a Digital Transmission System

There are two possible ways how the receiver passes on the information (obtained from the demodulated received sequence) to the decoder:

- The receiver decides for every received code symbol the corresponding code symbol that was most likely sent. This is then passed on to the decoder without sending any other information about the certainty or likelihood of the decision. This method of decoding is termed *hard decision decoding*. This means that for the previously mentioned time-discrete memory-less channel, the components of the error vector **e** and the received vector **r** have the same range of values as the code symbols. In this binary case this would translate to elements from GF(2).
- The decoder passes an additional value to the decoder. This additional value expresses the reliability of the preceding decision for a particular code symbol. With a binary code alphabet this would mean that the real value and not just a one or zero is passed on to the decoder. For the case of a hard decision decoding process, the real valued received vector would be transformed into the closest matching binary symbol, and passed on to the decoder. With *soft decision decoding* the decoder receives the real valued symbols, and can use this additional information, often referred to as channel information, in the decoding

process. The effect of soft decision decoding, is that for the same bit error rate, a low code rate code needs about 2 dB E_{bit}/N_0 less using this method, than with hard decision decoding. With higher code rates, this difference becomes smaller. It is intuitively assumed that the components of the error vector **e** and the received vector **r** are real valued.

Independent of which of the above mentioned schemes is used, the result of the decoding process can be grouped into either one of the following categories:

- *Correct Decoding*, which implies that the decoder has identified the errors introduced by the channel, and has corrected them. The decoder thus delivers the same information that was initially sent to the information sink.
- *Incorrect Decoding*, which means that the decoder does not deliver to the information sink the same information that was initially sent.
- *Decoding Failure*, implies that the decoder does not deliver any information vector to the information sink.

The following methods are envisaged in the decoding process:

- *Error Detection*
  Here the decoder only checks whether or not the received vector represents a valid codeword. If this is not the case, then the decoder does not perform any operation aimed at correcting such an inherently incorrect vector. In the case where $\mathbf{e} \in C$, the decoder is not even able to recognize the error seeing that $\mathbf{r} \in C$ even though $\mathbf{r} \neq \mathbf{c}$.
- Maximum Likelihood Decoding
  If the decoder makes a decision (for any given received vector r) about which one of the codewords $\hat{\mathbf{c}}$ was most likely sent, in other words,

$$P(\hat{c}|r) = \max_{c \in C} P(c|r) \tag{2.34}$$

then it is called the *Maximum-A-Posteriori* (MAP) decision rule. According to Bayes theorem, the following holds true:

$$P(c|r) = \frac{P(r|c) \cdot P(c)}{P(r)}. \tag{2.35}$$

Since we are dealing with the received vector **r**, the probability of it being the one out of all the possible codewords,

$$P(r) = \sum_{all\ c \in C} P(r|c) \cdot P(c) \tag{2.36}$$

is constant, as long as it is the vector under investigation. If the codewords **c** are all equiprobable (in other words, all the possible codewords *A-Priori-Probabilities* $P(c) = 1/q^k$ are the same), then the following decision rule applies:

$$P(r|\hat{c}) = \max_{c \in C} P(r|c) \tag{2.37}$$

If the decoding is done according to this principle, then it is termed *Maximum Likelihood* decoding.

- *Bounded Minimal Distance Decoding*
  In this method, not all the possible received vectors r are decoded, but only those in which a bounded amount of errors exist. A possible undesirable outcome of this type of decoding, is decoding failure.

For the different decoding methods [10], error probabilities can be defined as follows:

- *Block Error Probability*

$$P_{Block} = \lim_{\mu \to \infty} \frac{Number\ of\ incorrectly\ decoded\ codewords}{Number\ \mu\ of\ sent\ codewords} \tag{2.38}$$

- *Bit Error Probability*

$$P_{Bit} = \lim_{v \to \infty} \frac{Number\ of\ incorrectly\ decoded\ information\ bits}{Number\ v\ \ of\ sent\ information\ bits} \tag{2.39}$$

The optimal decoding (*i.e.*, the error probability is minimized) of a binary block code or convolutional code with the incorporation of reliability information, (*i.e.*,soft decision decoding) will be investigated in more detail for the BPSK modulation system transmitting over a AWGN-channel.

To derive an expression for the decoding strategy and the metric, the detailed system representation of **Figure 4** is used. The already described time discrete, memory-less channel adds to the transmitted sequence a real valued error vector **f**. This is not to be confused with the error vector *e* with components from *GF*(2). As explained in the definition of soft decision decoding, the channel decoder uses this real valued received sequence **y** for decoding, and also assumes the function of the modulation specific decoding.

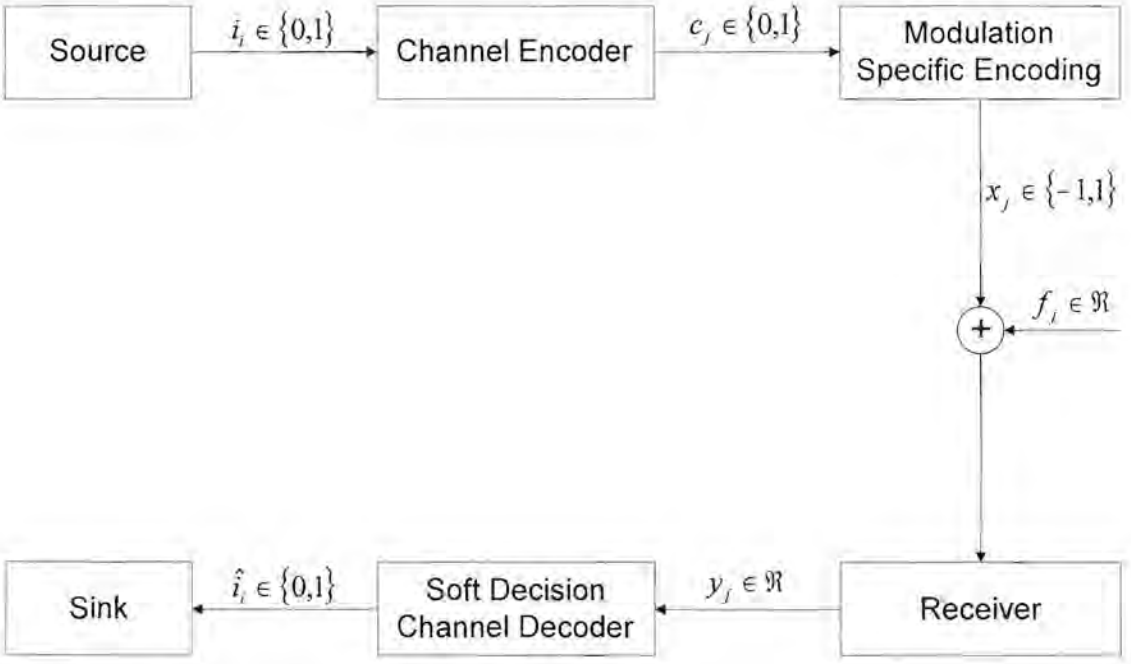

Figure 4 System Model for the Derivation of Decoding Strategy

The components f_j of the error vector **f** are (given the present assumptions) normally distributed random variables with the following density function:

$$\varphi\left(f_j\right) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{1}{2}\left(\frac{f_j}{\sigma}\right)^2} \tag{2.40}$$

with $\sigma^2 = N_0$ where N_0 is the single sided noise power spectral density. Analog to the previous discussion, the components y_i of the received vector **y** are also normally distributed random variables with average values of -1 and 1, dependant on the sent value x_j of the transmission sequence **x**. It follows that the density function of y_j is also dependant on the variable x_j.

$$\varphi\left(y_j\middle|x_j\right) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{1}{2}\left(\frac{y_j - x_j}{\sigma}\right)^2}. \tag{2.41}$$

The two density functions[10] $\varphi\left(y_j\middle|x_j = -1\right)$ and $\varphi\left(y_j\middle|x_j = 1\right)$ are shown in **Figure 5**.

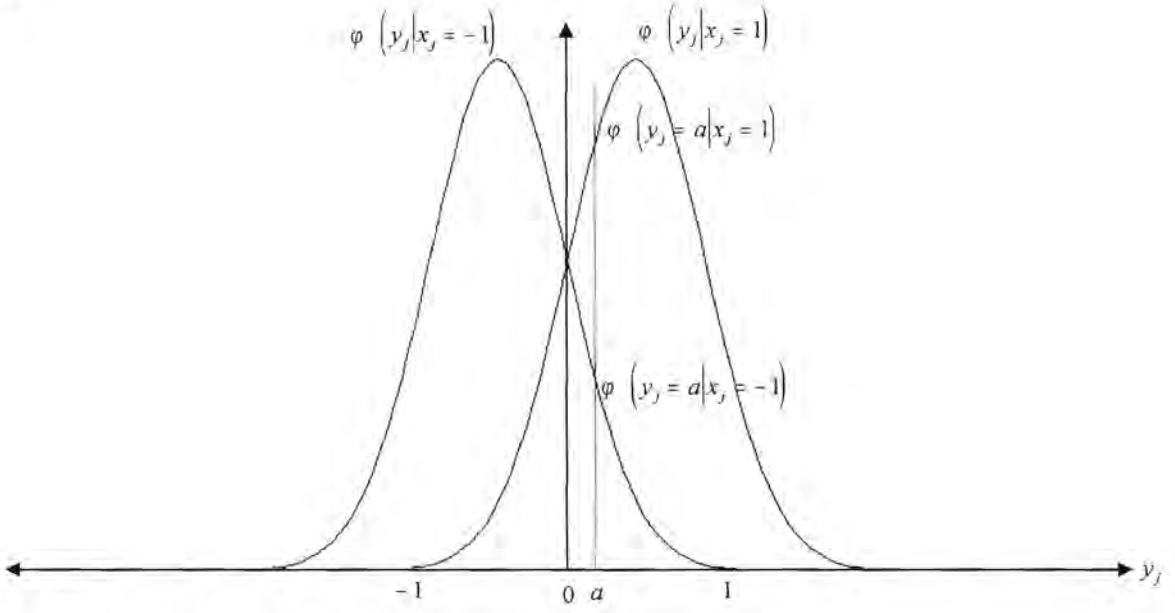

Figure 5 Probability Density Functions

Since f_j and y_j are stationary probability values (i.e. their density functions

$$\Phi\left(f_j\right) = \int_{s=-\infty}^{f_j} \varphi(t) \cdot dt \quad i.e. \quad \Phi\left(y_i \middle| x_j\right) = \int_{s=-\infty}^{y_j} \varphi\left(t \middle| x_j\right) \cdot dt \tag{2.42}$$

are stationary), the probability $P\left(y_j = a \middle| x_j\right)$ is zero for any arbitrary value of a.

For the purpose of the following discussion, the MAP decision rule of equation (2.34) is considered. If, instead of the received vector **r** with discrete values for the components r_j, the vector **y** with real valued components y_i is used, then equation (2.34) changes to:

$$P\left(\hat{x} \middle| y\right) = \max_{all\ x} P\left(x \middle| y\right). \tag{2.43}$$

Since, as explained above, no more discrete probabilities P(**y**|**x**) exist, the terms P(**r**|**c**) and P(**r**) from equation (2.34) have to be replaced with φ(**y**|**x**) and φ(**y**) respectively.

$$P\left(x \middle| y\right) = \frac{\varphi\left(y \middle| x\right) \cdot P(x)}{\varphi\left(y\right)} \tag{2.44}$$

where

$$\varphi\left(y\right) = \sum_{all\ x} \varphi\left(y \middle| x\right) \cdot P(x), \tag{2.45}$$

which for a given **y**, just as with P(r) from equation (2.34), results in a constant value. Due to the before mentioned fact, the decision rule as stated in equation (2.43) can be rewritten into the following decision rule:

$$\varphi\left(y \middle| \hat{x}\right) = \max_{all\ x} \varphi\left(y \middle| x\right) \tag{2.46}$$

This means, that the most likely sent vector $\hat{x}$ for equally likely transmitted vectors **x** is the one which maximizes the above probability density function.

The expression $\varphi(y|x)$ to be maximized, is transformed with the assistance of equation (2.41). The vectors **x** and **y** each have n components in this case.

$$\varphi(y|x) = \prod_{l=1}^{n} \varphi(y_l|x_l) = \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}\cdot\sigma} \cdot e^{-\frac{1}{2}\left(\frac{y_l - x_l}{\sigma}\right)^2} = \left(\frac{1}{\sqrt{2\pi}\cdot\sigma}\right)^n \cdot e^{-\frac{1}{2\sigma^2}\sum_{l=1}^{n}(y_l - x_l)^2} \quad \textbf{(2.47)}$$

To maximize this expression, it is necessary to minimize the sum in the exponential function.

$$\sum_{l=1}^{n}(y_l - x_l)^2 = \sum_{l=1}^{n}(y_l^2 - 2x_l y_l + x_l^2) = \sum_{l=1}^{n} y_l^2 - 2\sum_{l=1}^{n} x_l y_l + \sum_{l=1}^{n} x_l^2 \quad \textbf{(2.48)}$$

From the above expression it follows that $\sum y_l^2$ (given the received vector **y**) as well as $\sum x_l^2 = n$ are constant, which implies that in order to minimize the given sum, only $\sum x_l y_l$ has to be maximized.

In conclusion, it can be stated that with BPSK transmission of n code symbols over an AWGN channel, the Maximum-Likelihood-Decoder maximizes the following sum:

$$\sum_{l=1}^{n} x_l y_l = x \cdot y^T \quad \textbf{(2.49)}$$

In other words, the scalar product of the sent vector x and the received vector y is maximized by the Maximum-Likelihood-Decoder.

2.4.2 Decoding of Block Codes

For any given code, there usually exists a multitude of different decoding procedures [10], [1], [2]. The choice of any given decoding procedure above any other is usually governed by the type of application. Other factors which contribute to the choice are, for instance, the availability of computing power, the speed of the specific decoding and the ease of implementation of given algorithm. These factors obviously also have to be considered when choosing a code, since these factors will impose restrictions on the type of code to be used, and also limit the parameter choices for the code (e.g. the possible

values of n and k may be severely limited by the non-availability of decoding power).

If in addition, channel state information also has to be considered in the decoding process (Soft-Decision Decoding) and the decoding done via a Maximum-Likelihood process (the procedure will be termed SDML decoding in future), then there are only very few decoding procedures that allow for a practical decoding of given block code. Theoretically, every linear block code can be decoded by a SDML decoding process, in that every received vector is compared to all q^k possible sent vectors and the most likely one chosen as per equation (2.43). It can be seen that in a decoding procedure utilizing the above outlined method, q^k scalar products have to be calculated, and the maximum chosen. Since the number of calculations increase exponentially with k, it is not a very viable decoding procedure for codes with large k. The main topic of this work is to provide procedures that are able to provide more efficient decoding for high rate codes with small n-k. The following chapters will be dedicated to the development and fine-tuning of such procedures.

2.4.3 Decoding of Convolutional Codes

The fact that with convolutional codes the n code symbols are not only dependant on the k information symbols of the current information vector, but also on M previous information vectors, is the reason why this decoder has a lot more information available to do the decoding on than a block decoder. The above comparison holds true for all codes with comparable parameters such as n and k. This then also explains why the decoding results for memory-less channels obtained by the use of a convolutional decoder are superior to those obtained with block decoders.

In order to perform SDML decoding on a convolutional code with a code set C of T code blocks, it is necessary to compare this with all the possible $q^{T \cdot k}$ code sequences as describes in equation (2.43). Viterbi showed in 1967 that in the Trellis diagram of **Figure 2**, it is not necessary to compare all qk paths with the received vector, but that it is sufficient, at any given time instant i, to only consider the incoming path into a state having the "best" metric. By applying equation (2.49) the assumption is made that the decoder is in a state which best corresponds to the most likely sent code sequence. This path is termed the survivor path.

The following figure outlines the process proposed by Viterbi [14].

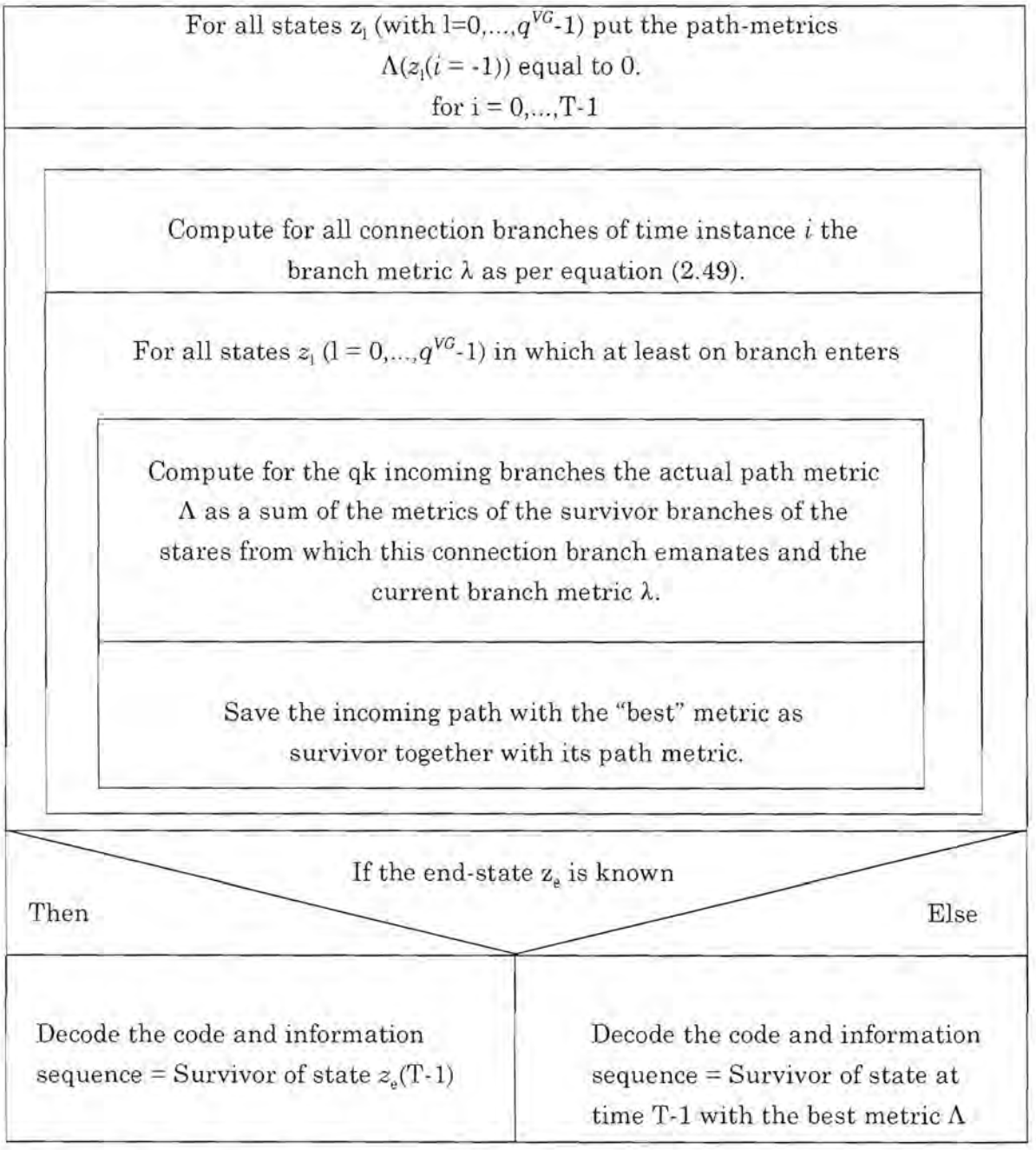

Figure 6 Schematic Representation of the Viterbi Algorithm

In the next chapter, the principles given are used in order to provide SDML methods for the decoding of block code trellises.

Chapter 3

Decoding of Block Codes

3.1 Introduction

In the previous chapter it was explained how every linear block code can theoretically be Soft-Decision-Maximum-Likelihood decoded (SDML-decoded) by comparing the received vector with all possible q^k sent vectors, and the most likely one chosen. In this chapter the two most popular non-algebraic decoding methods for block codes are presented. The first one considered is the method suggested by Bahl, Cocke, Jelinek and Raviv in 1974 [2], and the second one the procedure described and outlined by Wolf [16]. Both methods reduce the number of comparisons required dramatically. In **Section 3.1**, the term *Code Trellis Diagram* [5] is introduced and explained. A special case of the code trellis diagram is termed the *Syndrome Trellis Diagram* [5], the construction of which is described in **Subsection 3.2**. In the above mentioned code trellis diagram SDML-decoding can be performed by applying the Viterbi algorithm as described in the last section of **Chapter 2**. Firstly however, the method to obtain the parity check matrix **H**, required to construct the code trellis diagram from the generator matrix **G**, is explained in **Subsection 3.3**.

Throughout the chapter, a number of practical examples are given to illustrate various aspects of block code trellis construction.

3.2 Block Codes

A true code trellis (also termed a true code trellis diagram) of a code C is a trellis diagram,described in **Subsection 2.3.3**, with the property that a unique mapping between the codewords of the code C and the paths of the trellis diagram exists. That implies that for the case of a binary linear block code, as described in this section, the following holds true:

- The trellis diagram has a starting state (states will from time to time also be described as nodes) z_0 (i = -1) and an ending state z_e ($i = n$ - 1).
- Each connecting branch is assigned a binary value (i.e. a code bit), under the condition that two branches leaving the same state or node cannot have the same binary value assigned to them.
- A path from z_0 (i = -1) to z_e ($i = n$ - 1) with assigned code bits $c_0, c_1, \ldots, c_{n-1}$ exists when $c = (c_0, c_1, \ldots, c_{n-1}) \in C$.

The term "*State*" means exactly the same as the term "*Node*", and the naming of states and their ordering into a plane of states is completely arbitrary. In **Figures 7** and **8**, two possible "*Untrue*" trellis diagrams are shown, for the (3,2)-Parity Check Code with generator matrix:

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{3.1}$$

The reason why they are called "*Untrue Trellis Diagrams*" is the fact that the second condition listed above is not met for each of the first nodes.

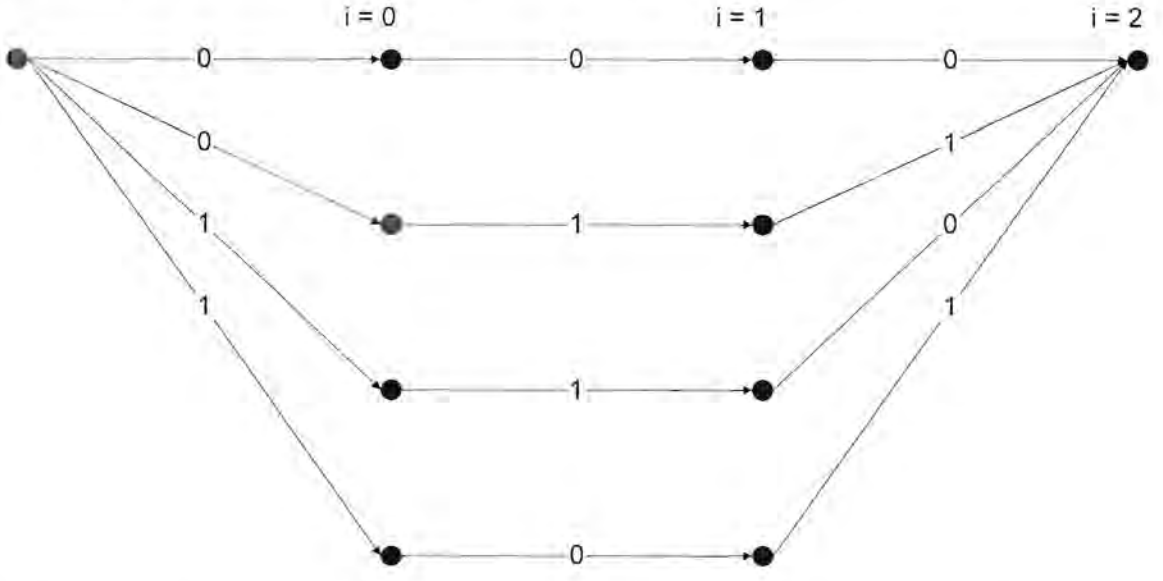

Figure 7 Untrue Trellis Diagram A for the (3,2)-Parity-Check-Code

It may never happen that after stepping through the trellis diagram one code symbol at a time one has to traverse backwards in order to find the correct path. The trellis diagram in **Figure 7** is called a trivial trellis diagram of a block code.

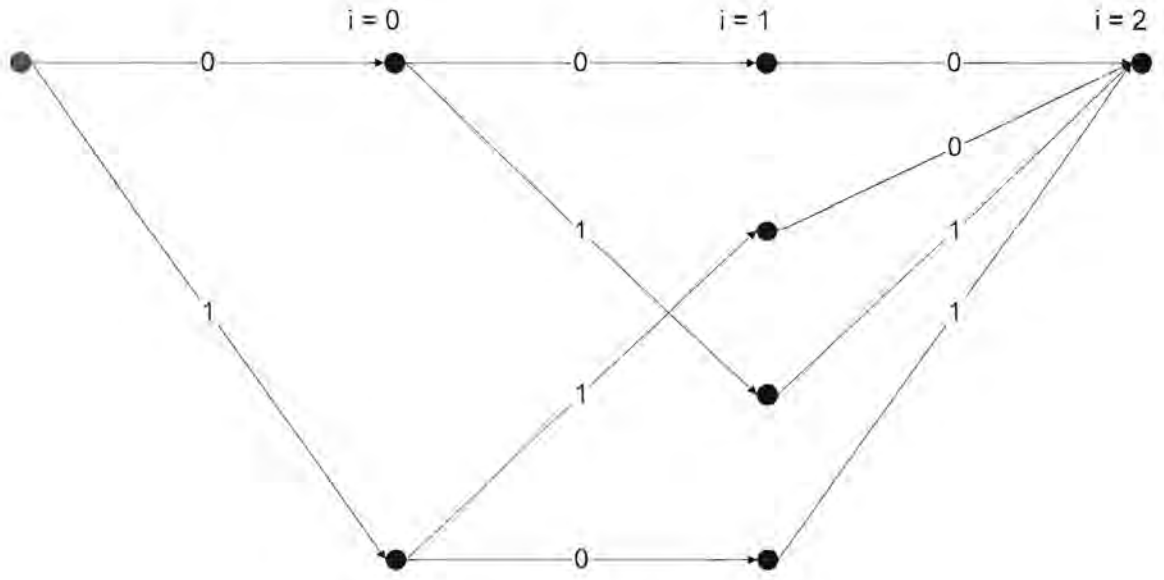

Figure 8 True Trellis Diagram A for the (3,2)-Parity-Check-Code

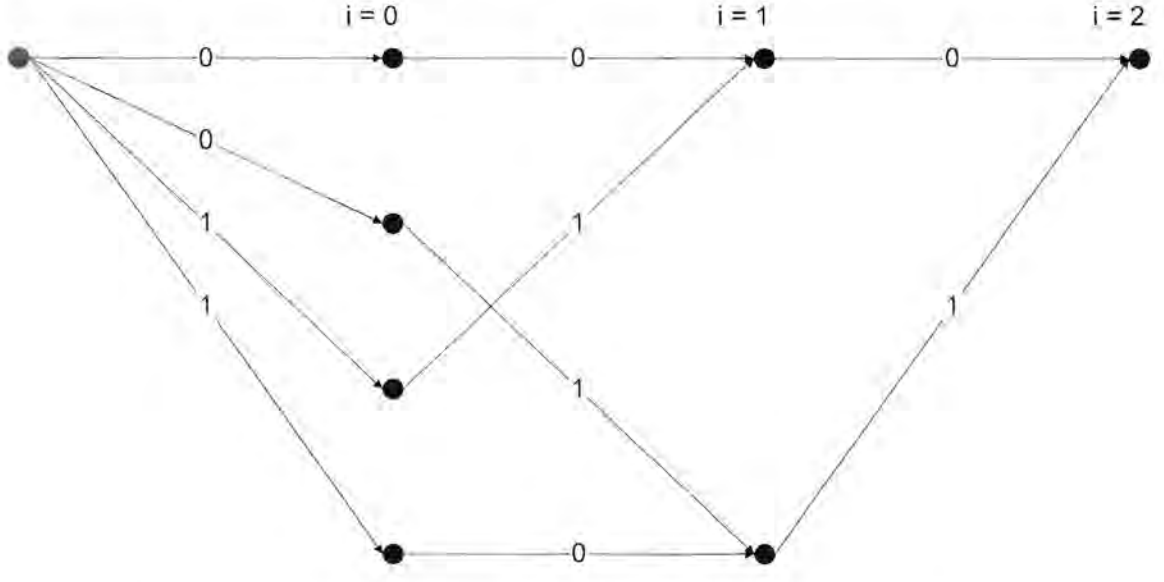

Figure 9 Untrue Trellis Diagram B for the (3,2)-Parity-Check-Code

Two code trellis diagrams are "*Isomorph*" if the one can be transformed into the other by changing some or all planes of states. Examples of isomorph trellises are shown in **Figures 10** and **11**, both of which are also valid or true code trellis diagrams.

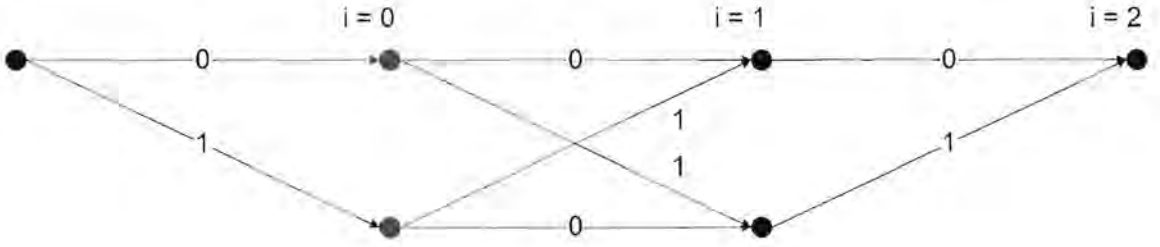

Figure 10 True Trellis Diagram C for the (3,2)-Parity-Check-Code

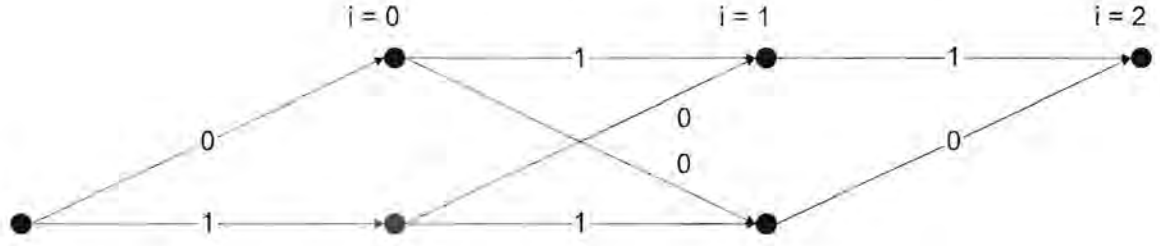

Figure 11 True Trellis Diagram B for the (3,2)-Parity-Check-Code

A code trellis is termed minimal, if for all planes of states the number of nodes in the given trellis is minimal compared to all other possible code trellis diagrams for the specific code C. Both the code trellis diagrams in **Figures 10** and **11** are minimal code trellis diagrams for the (3,2)-Parity-Check-Code. This is true, since there exists no other code trellis diagram for the same code C in which the planes $i = 0$ and $i = 1$ deliver less nodes. Furthermore, a minimal code trellis diagram for a code C is unique except for the isomorph variations in the trellis diagrams of the same code C. This means that two minimal trellis diagrams for the same code C which are not also isomorph cannot exist.

Of utmost importance for the following sections, is the fact that the Viterbi Algorithm described in **Subsection 2.4.3** can now be applied to a true code trellis diagram for the SDML decoding of block codes, without the necessity to compare the received vector with all of the possible q^k sent vectors.

3.3 Syndrome Trellis

A linear block code is specified through its parity check matrix [10] $\boldsymbol{H} = \left(h_{ij}\right) = \left(h_1, h_2, \ldots, h_n\right)$ as described in **Section 2.1.1**. **Equation 2.5** can be rewritten in the following manner:

$$\boldsymbol{c} \cdot \boldsymbol{H}^T = \boldsymbol{c}_0\boldsymbol{h}_1^T + \boldsymbol{c}_1\boldsymbol{h}_2^T + \ldots + \boldsymbol{c}_{n-1}\boldsymbol{h}_n^T = \boldsymbol{0} \tag{3.2}$$

or

$$\boldsymbol{c}_0\boldsymbol{h}_1 + \boldsymbol{c}_1\boldsymbol{h}_2 + \ldots + \boldsymbol{c}_{n-1}\boldsymbol{h}_n = \boldsymbol{0}^T \tag{3.3}$$

where $\mathbf{0}$ is the zero row vector of length n - k and $\mathbf{0}^T$ the corresponding column vector.

A state $z^T(i)$ is defined as

$$z^T(i) = \boldsymbol{c}_0\boldsymbol{h}_1 + \boldsymbol{c}_1\boldsymbol{h}_2 + \ldots + \boldsymbol{c}_i\boldsymbol{h}_{i+1} = \sum_{p=0}^{i} \boldsymbol{c}_p\boldsymbol{h}_{p+1} \tag{3.4}$$

where $z^T(i)$ (as $\mathbf{0}^T$) is a column vector with n - k components.

From this, it is possible to construct a valid code trellis diagram, termed a *syndrome trellis*, diagram defined by:

$$z^T(i-1) + \alpha \cdot h_{i+1} = z^T(i) \tag{3.5}$$

Equation 3.5 is interpreted as follows:

- At the time instance i = -1 only one state exists. This state is called the starting state $z_0^T(i = -1) = 0^T$.
- For all i = 0, 1, ..., n - 1:
  - $z^T(i)$ is exactly then a state if at time i a connecting branch enters and for any $\alpha \in$ GF(q). This means that the number of states at time i is computed by inserting all $\alpha \in$ GF(q) into all states $z^T(i$ - 1) of **Equation**

3.5.

- At the same time a connection branch is constructed from the state $z^T(i-1)$ to the corresponding state $z^T(i)$ and labeled $\alpha = c_i$, according to **Equation 3.5.**

- Finally, all nodes without a path ending in the final state $z_e^T(1=n-1)=0^T$ are removed as well as all other nodes which branch off these removed nodes. This can be seen from **Equations 3.3** and **3.4**. This expurgating or "cleaning" of the code trellis diagram is not absolutely necessary for the decoding process. The decoding can also be done on a non-expurgated trellis, but then only paths which do end in the state $z_e^T(1=n-1)=0^T$ must be considered, and the others ignored.

The syndrome trellis diagram obtained by means of the above method consists of a total of $n+1$ planes each comprising a maximum of q^{k-n} states. The q^k paths of the trellis diagram starting at state $z_0^T(i=-1)=0^T$ and ending in state $z_e^T(1=n-1)=0^T$ represent the q^k codewords.

Since the syndrome trellis diagram only contains q^k paths, another upper bound for the maximum number of states in a plane exists, since at any time i no more than q^k states can be reached by traversing q^k paths. This then defines the upper bound on the number of states Ni at time i as:

$$N_i \leq q^{\min(n-k,k)} \tag{3.6}$$

In order to illustrate the process outlined above, the syndrome trellis diagram for the (7,4)-Hamming-Code is constructed. The parity check matrix for the above mentioned code is as follows:

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Beginning at the starting state

$$z_0^T(i=-1) = 0^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.8}$$

two connecting branches, one for $\alpha = 0$ and one for $\alpha = 1$ respectively are constructed to the states

$$z^T(i=0) = z^T(i=-1) + 0 \cdot \boldsymbol{h}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + 0 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.9}$$

and

$$z^T(i=0) = z^T(i=-1) + 1 \cdot \boldsymbol{h}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}. \tag{3.10}$$

According to **Equations 3.9** and **3.10**, at time $i = 0$ the states mentioned above are possible. Which one of them actually occurs is dependant on the first code bit c_0.

At each one of these two states, two new connecting branches emanate. In order to determine the two new destination states, the following two expressions are added to $z^T(i = 0)$:

$$0 \cdot \boldsymbol{h}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \text{and} \qquad 1 \cdot \boldsymbol{h}_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

For each level or plane in the syndrome trellis diagram the process is repeated for each active or connected node. The correct column vector of the parity check matrix has to be used for every iteration.

The result of this iterative construction is depicted in **Figure 12**. The solid lines depict the case for which $\alpha = 0$ and the dotted lines the case $\alpha = 1$. It has to be noted that when

$\alpha = 0$, the trellis will always move from a certain state j in plane m to the same state k in plane n. This allows for fast trellis construction, when the system is implemented in hardware, as half of the connecting branches are calculated by default. This saves a tremendous amount of calculating time, which in term means more effective transmission systems.

If the trellis is expurgated as described above, a trellis diagram representation is obtained as depicted in **Figure 13**.

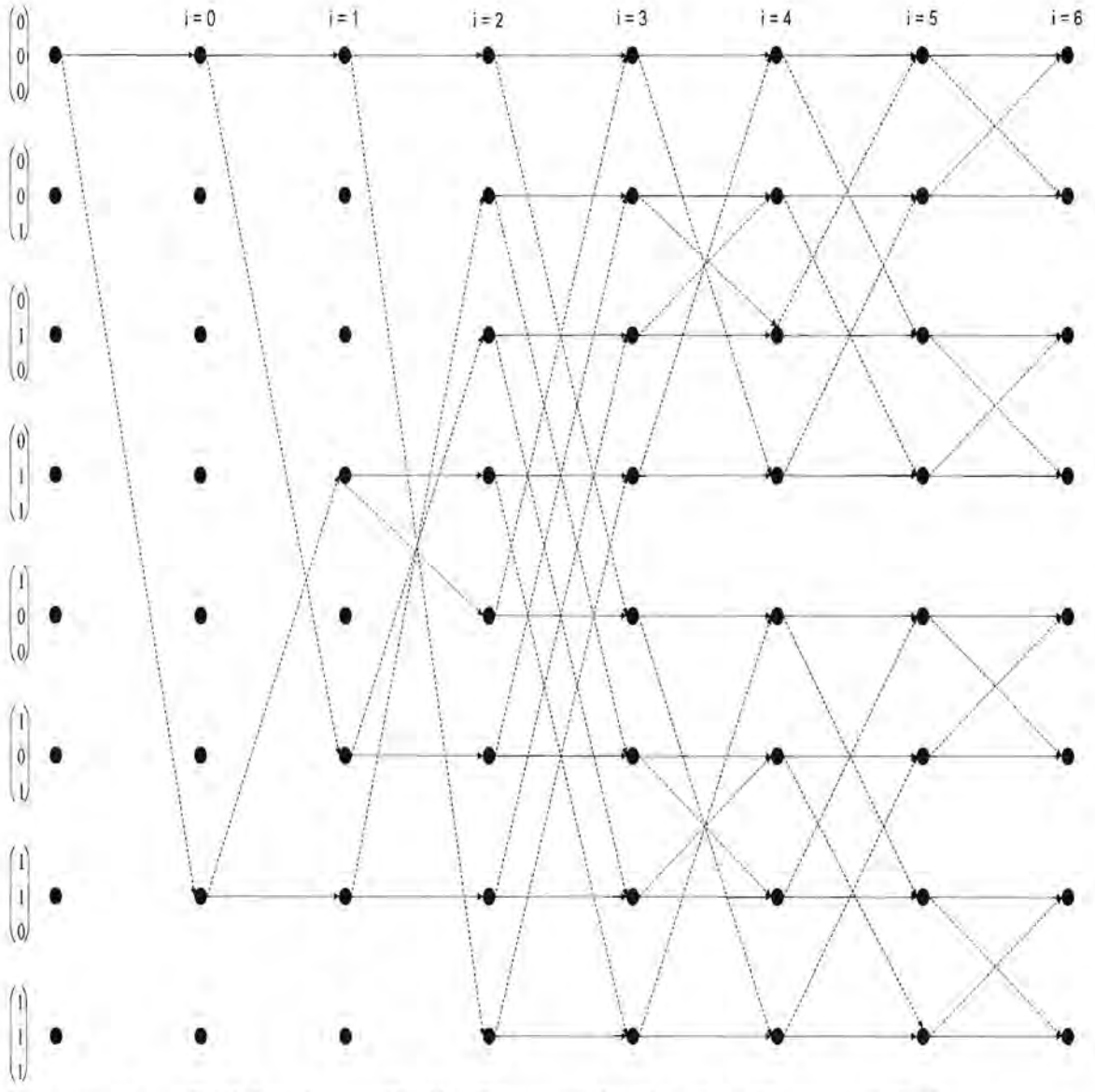

Figure 12 Full Syndrome Trellis Diagram for the (7,4)-Hamming-Code

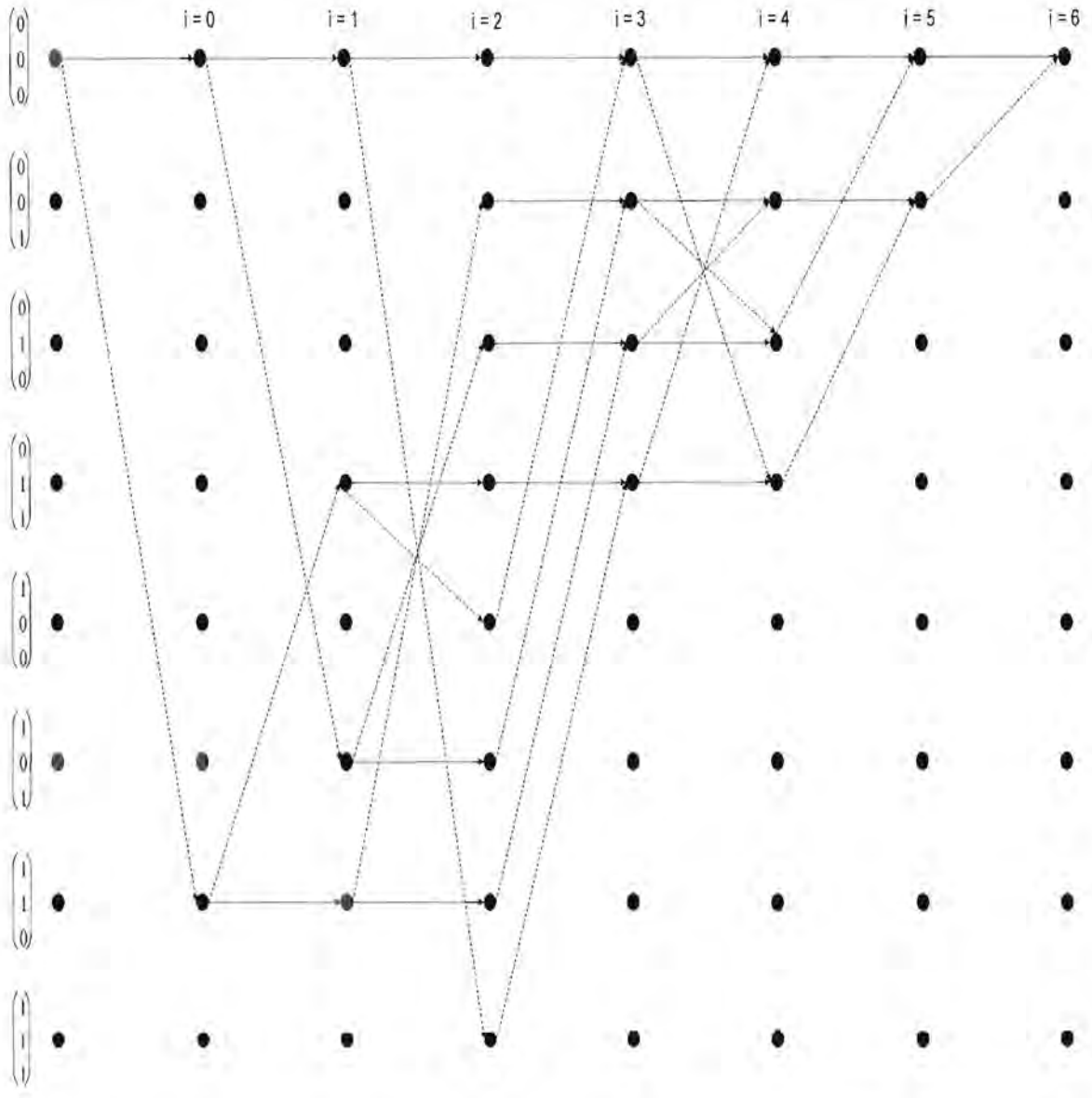

Figure 13 Expurgated Trellis Diagram for the (7,4)-Hamming-Code

As shown in **Section 3.2**, it is indeed possible to SDML decode a block code by utilizing its syndrome trellis diagram (a special case of a true code trellis diagram). The complexity of the decoding is dependant on the number of nodes in the trellis. This is governed by the following equation:

$$N_{\Sigma} = \sum_{i=-1}^{n-1} N_i \tag{3.11}$$

From the definition of the minimal code trellis diagram in **Section 3.2**, it follows that a code trellis diagram is minimal when N_{Σ} is minimal for all code trellis diagrams of C. The syndrome trellis diagram of a given linear block code is a minimal code trellis diagram. This means, that there exist no other non-isomorph code trellis diagrams with fewer nodes than the code trellis diagram obtained through the use of the syndrome construction procedure outlined above.

However, this does not hold for equivalent codes. As was shown in **Subsection 2.2.2**, an equivalent code can be obtained from a code C, by performing matrix operations on the parity check matrix of the original code C. For such an equivalent code it is very possible and also likely, that syndrome trellis diagrams can exist which have a different number of nodes. An example for such a case is the (5,3)-Code with the following parity check matrix.

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.12}$$

The syndrome trellis diagram for this code is depicted in **Figure 14**.

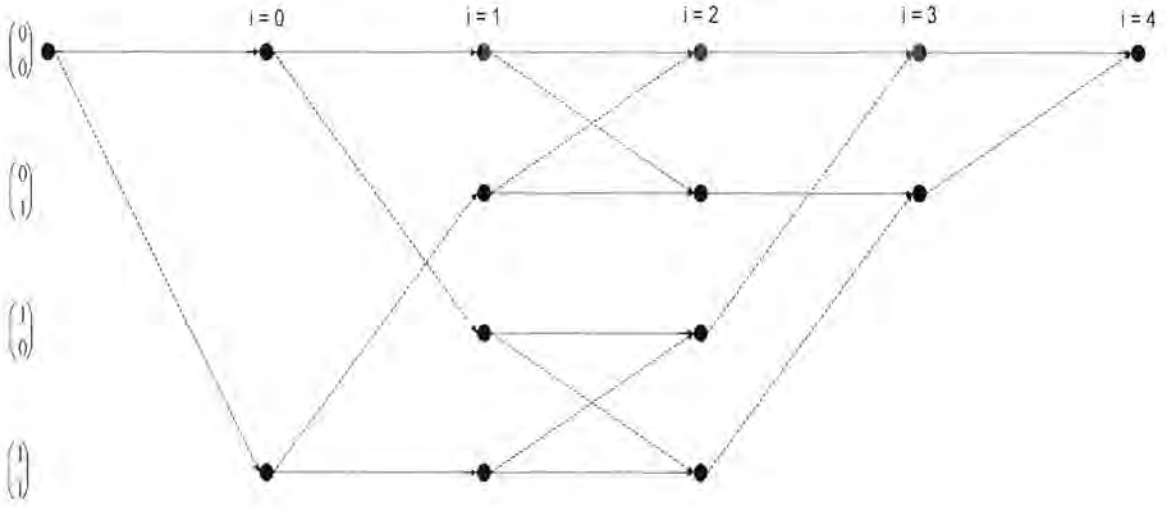

Figure 14 Syndrome Trellis Diagram A for the (5,3)-Code

If columns of the parity check matrix of **Equation 3.12** are interchanged, an equivalent code is obtained with the following parity check matrix.

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{3.13}$$

The syndrome trellis diagram for this code is depicted in **Figure 15**.

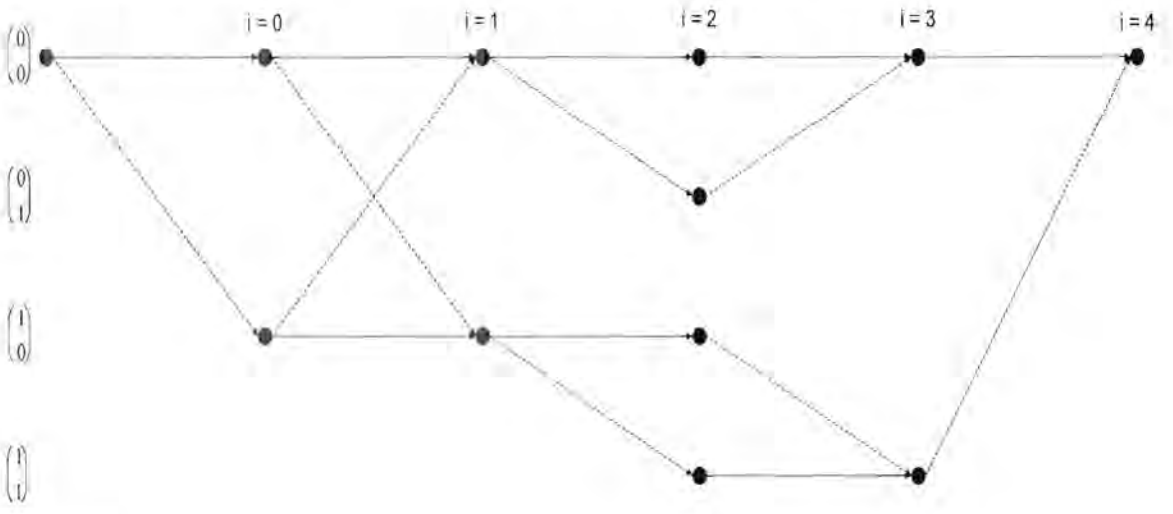

Figure 15 Syndrome Trellis Diagram B for the (5,3)-Code

As can be seen from **Figures 14** and **15** above, the number of nodes differ for the two equivalent codes. In **Figure 14**, N_{Σ} equals 14, but in **Figure 15**, N_{Σ} only equals 12. It can be seen that in both cases $q^k = 8$ paths traverse the code trellis diagrams. These 8 paths represent the $q^k = 8$ codewords. In the second diagram however, the topological distribution of these paths is more favorable than the distribution in the second diagram. The reason for this being that the reduced number of nodes imply a much simpler decoding complexity. To summarize, it can be said that the less nodes a syndrome trellis diagram contains, the simpler the eventual decoding becomes.

For cyclic codes an alternative code trellis diagram construction method exists, which differs completely from the syndrome trellis diagram construction technique described previously. It is done by using the content of the shift registers (which are used in the coding and decoding process) as states in the trellis diagram. This method will however not be considered here, as it does not produce a minimal trellis diagram. Another

method of obtaining a syndrome trellis diagram is considered in **Appendix 1**, where it is included as an example.

As mentioned in **Section 3.3**, the decoding complexity relates to the number of nodes in the code trellis diagram. It should be clear that a good trellis construction is the procedure that will result in a trellis being minimal, i.e. the trellis should have the least number of nodes N_{Σ} possible. This is indeed an immense task, since $n!$ permutations of the column vectors of the parity check matrix will have to be processed in order to find the equivalent code trellis diagram with the least amount of nodes, i.e. the minimal trellis representation. For smaller codes, this process can be done by hand, but for a large code ($n>>$), this task becomes nearly intractable.

In a following chapter, methods are investigated which reduce the number of nodes in the code trellis diagram.

3.4 Derivation of the Parity Check Matrix

In **Section 3.3** the parity check matrix **H** was assumed known for the calculation of the syndrome trellis diagram [16]. A decoder would normally only have the generator matrix **G** available for the specific code to be decoded. In order to perform the decoding, the decoder will have to determine the topological structure of the trellis diagram, for which it will need to have the parity check matrix **H**, obtainable from the generator matrix **G**.

For generator matrixes in the systematic form **G'** defined in **Equation 2.7**, it is fairly elementary to find the parity check matrix according to **Equation 2.9**. For the general non-systematic form of the generator matrix, this process is however a bit more complicated.

A method that presents itself, is the direct solving of **Equation 2.6** ($\mathbf{G}\cdot\mathbf{H}^T = \mathbf{0}$) after fixing the values of $(n - k)^2$ unknowns. This approach is not a trivial process since it is not just a matter of fixing any of the $(n - k)^2$ unknowns, but also which of the unknowns to fix. This is explained in more detail in the following outline:

- For non-systematic block codes, it is also possible to obtain a matrix representation in the form of **Equation 2.9**. The $(n - k)^2$ chosen unknowns would

be the elements of the ((n - k) × (n - k)) unit matrix in $\mathbf{H}' = (-\mathbf{A}^T|\mathbf{I}_{n-k})$. If the given generator matrix G is split up into the two sub-matrixes $\mathbf{G}_{left}$ and $\mathbf{G}_{right}$ where $\mathbf{G}_{left}$ is a (k × k) matrix and $\mathbf{G}_{right}$ a (k × (n - k)) matrix, i.e. $\mathbf{G} = [\mathbf{G}_{left} \mid \mathbf{G}_{right}]$, then **Equation 2.6** can be rewritten as

$$\boldsymbol{G} \cdot \boldsymbol{H}'^{T} = \left(\boldsymbol{G}_{left} \middle| \boldsymbol{G}_{right}\right) \cdot \begin{pmatrix} -\boldsymbol{A} \\ \hline \boldsymbol{I}_{n-k} \end{pmatrix} = -\boldsymbol{G}_{left} \cdot \boldsymbol{A} + \boldsymbol{G}_{right} = \boldsymbol{0} \tag{3.14}$$

or

$$\boldsymbol{G}_{left} \cdot \boldsymbol{A} = \boldsymbol{G}_{right} \tag{3.15}$$

The following are the n - k linear equations for the calculation of the n - k column vectors of the matrix $\boldsymbol{A} = (a_1, a_2, \ldots, a_{n-k})$:

$$\begin{aligned} \boldsymbol{G}_{left} \cdot a_1 &= g_{right,1} \\ \boldsymbol{G}_{left} \cdot a_2 &= g_{right,2} \\ &\vdots \\ \boldsymbol{G}_{left} \cdot a_{n-k} &= g_{right,n-k} \end{aligned} \tag{3.16}$$

Each of these (inhomogeneous) equations only has a solution if the rank of the extended matrix $[\mathbf{G}_{left} \mid \mathbf{G}_{right,i}], i = 1,2,\ldots,n - k$, is not larger than the rank of $\mathbf{G}_{left}$. This is however not necessarily fulfilled, since the first k columns of the matrix G, which forms the matrix $\mathbf{G}_{left}$, can be linearly dependant, which in turn could make the rank of $\mathbf{G}_{left}$ smaller than k. By extending the matrix $\mathbf{G}_{left}$ by adding one column from $\mathbf{G}_{right}$, the rank can be increased but then the equations are not uniquely solvable.

This shows that it is not possible to choose and fix any of the $(n - k)^2$ unknowns.

- A possible solution to **Equation 2.6** can be found from a matrix $\widetilde{\boldsymbol{H}}$ with n - k

identical row vectors identical. This can be any particular code vector from $C^{\perp}$. $(n - k)^2$ unknowns can be chosen and fixed in such a manner that the matrix $\widetilde{H}$ solves **Equation 2.6**. However, the matrix obtained by the procedure above, is indeed not a parity check matrix, since it has the rank 1 and not $n - k$, which does not comply with the definition of a parity check matrix given in **Section 2.2**.

Since this approach does not render a tractable solution, the parity check matrix $\widetilde{H}$ has to be calculated on the following basis:

- Firstly the generator matrix **G** is, as described in **Section 2.2**, is transformed by elementary row matrix operations so that unit vectors of length k are present in k of the n columns of the matrix.
- By swapping columns, the transformed generator matrix is changed into its systematic form **G'** given by **Equation 2.7**.
- The corresponding parity check matrix **H'** is then determined from **Equation 2.9**.
- The column swapping procedure in step two above is then applied in reverse order on **H'** to so obtain a possible parity check matrix **H**.

An example of this procedure is given in **Appendix B**.

3.5 Decoding of the Trellis Diagram

As already mentioned several times, the Viterbi algorithm outlined **Figure 6** of **Subsection 2.4.3**, can be used to theoretically SDML-decode any linear block code through its syndrome trellis. This means that for every codeword C all the steps depicted in **Figure 7** have to be performed once with the end state $z_e^T(1 = n - 1) = 0^T$ known.

However, unlike the normal Viterbi algorithm the most likely sent codeword $\hat{c}$ and not

the most likely sent information vector **i** is obtained. The output is a valid codeword that satisfies **Equation 2.5**. Therefore, all that needs to be done is to reverse the mapping between the information vector and the code vector to obtain the most probable transmitted information vector. This is done with a lookup table or by solving **Equation 2.1**. In order for the decoder to work on an arbitrary block code, of which only the generator matrix is know to the decoder initially, the second method of obtaining the information vector from the code vector is preferred.