On Trellis Structure of Error Correction Coding

S.K.Thomas and M.A.Ambroze

Centre for Security, Communications and Network Research,
Plymouth University, Plymouth, UK
e-mail: info@cscan.org

Abstract

The major objective of all communication techniques is accomplishing the fastest way of data transmission through a communication channel, i.e. trying to reach the Shannon's Channel Limit. One of the solutions to these developments is error correction coding technique which is currently employed in broadband satellite communication and data storage. Constructing a Trellis, which is the graphical representation of the code; reduces the decoding complexity, thereby improves transmission efficiency. This project will investigate the way trellises are constructed for different types of codes, mainly linear block codes and how they are used to correct errors on transmission channels.
The project involves in the study of the trellis diagrams for both the convolutional and block codes; and focuses on the encoding and decoding of linear block codes using the trellis diagram. The implementation of the trellis diagram of the Hamming code for both the encoding and decoding process has been done using the MATLAB software. The code has been tested for various codewords and the results are collated in tables.

Keywords

Error Correction codes, Trellis representation, Convolutional coding, Linear Block codes, Hamming code, Hard decision Viterbi decoding, Soft Decision Viterbi Decoding

1 Introduction

A basic block diagram of a communication system is illustrated below. The information source can be either analog source or digital source. The analog source of information needs to be converted to digital bits for efficient transmission and this can be done using a sampler and analog to digital converter. In order to represent the digital information using the smallest number of bits, techniques such as removal of redundancy is used. The conversion of analog data into digital information efficiently is broadly classified as source coding. The channel encoder prepares the data from the source encoder for digital modulation and efficient transmission. The modulator matches the output of the channel encoder to the transmission channel. In the receiver section, the vice versa is performed to the received data to obtain the original data with minimum errors (Michelson and Levesque, 1985).

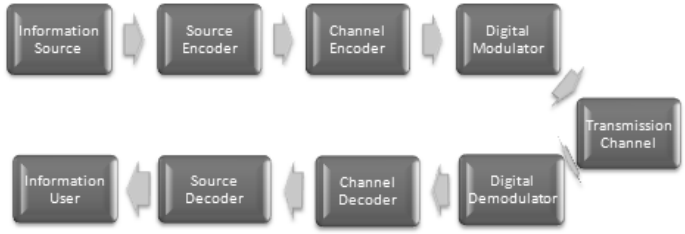

Figure 1: Block Diagram of a Digital Communication System (Michelson and Levesque, 1985)

The concept of error correction coding was introduced to minimise the errors occurring during the transmission of data and to recover the original data with minimum errors. In order to transmit information reliably, the information rate must be less than the channel capacity, and this was stated by Shannon's noisy coding theorem. It states that " It is theoretically possible to transmit information through a noisy channel with arbitrarily small probability of error provided that the information or source rate, R, is less than the channel capacity, that is R<C for reliable transmission"(Wade, 2000).

Error control coding is a practical method of achieving very low bit error rate after transmission over a noisy, band limited channel. An overview of error correction coding can be obtained in the following section.

2 Convolutional Codes

Convolutional coding is a method of channel coding where the check bits are periodically inserted in a continuous data stream.

2.1 Classifications:

Recursive Encoders – In this encoder, the memory bits gets added up and is connected with a feedback root.

Non – Recursive Encoders – In this encoder, the memory bits are added up without any feedback.

Systematic Encoder – A systematic code is one in which the original information bits can be identified.

Non – Systematic Encoder – In these codes, the information bits cannot be identified properly (Sankar, 2009).

Figure below shows a simple convolutional encoder. The information bits are passed into the encoder in small groups of k-bits at a time. The output bits are obtained by

performing modulo 2 addition (Exclusive OR operation) on the information bits and also the previous inputs.

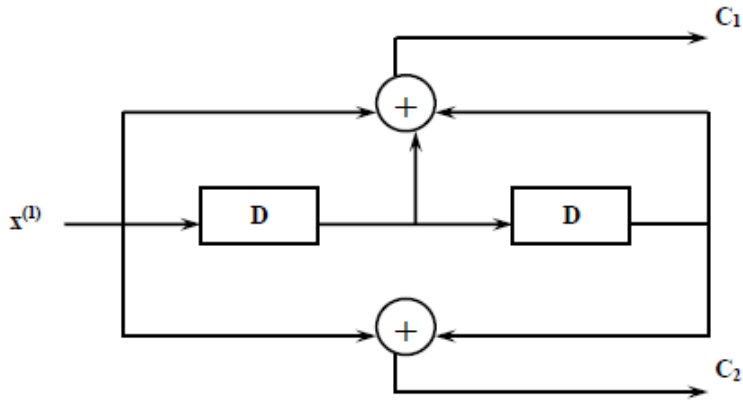

Figure 2: Convolutional Encoder with k=1, n=2 and r=1/2

The code rate R is expressed as R = k/n if the output of the encoder is n bits for every k input bits. In Figure, the value of k and n are 1 and 2 respectively. The constraint length of the code K is defined as the number of output bits affected for each information bit inputted into the encoder. In the above example, the value of K is 3.

All the shift registers are refreshed to a value of 0 before the encoding operation begins. For an input sequence of 01011, the encoded output will be 00 11 10 00 01(IIT, 2010).

3 Block Codes

Block code is the basic type of channel coding in which it adds redundancy to the message so that at the receiver end the decoding is done with minimal errors provided the information rate do not exceed the channel capacity. It contains a set of fixed-length vectors called code words. The main characteristic of block code unlike Huffman coding or Convolutional coding is that it is fixed length code words (Wade, 2000).

The block code has a set of fixed length vectors called code words whose length (n) is the number of elements in the vector. For a code word the elements are selected from an alphabet of q elements. If the alphabet has two elements 0 and 1, then it is a binary code and the elements are called bits. If the elements of a code are selected from an alphabet having q elements and if q>2 then the code is non binary. When q is a power of 2 i.e. q=2b (b is a positive integer), each of the q-ary element has got an equivalent binary representation which consists of b bits. Thus a non binary code having a block length N can be mapped into a binary code having a block length n=bN.

For a binary code of length n there are 2^n possible code words. From these code words we select M= 2^k code words in order to form a code. Thus we can say that a block of k information bits is mapped into a code word of length n which is in turn selected from a set of 2^k code words. The resulting block code is referred as an (n, k) code. The ratio k/n=RC can be defined as the rate of the code. The code rate parameter RC is simply the weight of the code word i.e. the number of non zero elements that it contains. Each code word has got its own weight and for a code the set of all weights constitutes the weight distribution. If all the M code words have equal weight then the code is considered as a fixed weight code or a constant weight code (Proakis, 2000).

For a digital communication we mostly use 0 and 1, the addition and multiplication is as shown below.

0+0= 0	0.0= 0
0+1= 1	0.1= 0
1+0= 1	1.0= 0
1+1= 0	1.1= 1

The multiplication and addition shown above are known as modulo-2 addition or multiplication and we can say that it is almost same as the ordinary arithmetic in which 2 is equal to 0. The symbols used here i.e. 0 and 1 along with the modulo-2 addition and multiplication can be termed as the field (binary field) of two elements. This is usually represented as GF (2) (Lin, 1970).

3.1 Hamming codes

Hamming codes have both binary and non binary properties but we consider only the binary properties. Binary hamming codes comprise a class of codes which follows the property

$$(n, k) = (2m – 1, 2m – 1\text{-}m)$$

Where

m= is any positive integer (i.e. if m=3 then we have (7, 4) code).

The parity check matrix H of the hamming code has a particular property. We have already mentioned in the previous section about the rows and columns of an (n, k) code, i.e. there are n- k rows and n columns for a (n, k) code. So when we consider a binary (n, k) hamming code the n= 2m- 1columns consists of every possible binary vectors with n- k =m elements and except all the zero vectors.

If we want to make a systematic hamming code the parity check matrix H can be arranged in the form below easily

$$H = [-P \vdots I_{n-k}]$$

From this the equivalent generator matrix G can also be obtained.

No two columns of the parity check matrix are linearly dependant or otherwise the two columns will be exactly the same or identical. But we can assume that if m>1, we can find three columns of the parity check matrix which adds to zero. So the minimum distance d_{min} will be equal to 3 for an (n, k) hamming code. A hamming code may also be shortened i.e. it can be made as (n-l, k-l). This is done by removing l rows from the generator matrix or by removing l columns from the parity check matrix.

Hamming distance is the count of the number of places in which each codeword differs from the hard decided received vector. The minimum distance dmin of a code is defined as the minimum Hamming distance between any two codewords of the code.

For any code with the minimum Hamming distance dmin, the number of errors that the code can detect is dmin – 1 and the number of errors it can detect is $\frac{d_{min}-1}{2}$.

For Hamming codes, the minimum Hamming distance dmin = 3 and therefore, it can detect 2 errors and correct 1 error.

In order to correct an error pattern, the receiver calculates the product:

$$S = Hr'$$

where r = c + e is the received vector and e is the error pattern. The value S is called the syndrome of the error and it is 0 if e = 0. If the value of S is a non-zero value, it shows that an error has occurred in the channel and $e \neq 0$. In general case, S is a column vector with N – K rows, corresponding to the N – K parity check equations of the code and it can take $2^{N-K} - 1$ non-zero values. If a code can correct t errors, then it has to have enough distinct syndromes to uniquely identify all possible error patterns of up to t errors (Ambroze, 2007).

The Hamming bound or Sphere packing bound for hard decision decoding is defined as:

$$\sum_{i=1}^{t} \binom{N}{i} \leq 2^{N-K} - 1$$

The error correction codes that satisfy this equation with equality are known as perfect codes.

For Hamming codes, dmin = 3, the number of errors it can correct t:

$$t = \frac{d_{min} - 1}{2} = \frac{3 - 1}{2} = 1$$

Let us consider the (7, 4) Hamming code for example where N = 7 and K = 4.

We have,

$$\sum_{i=1}^{t} \binom{N}{i} \leq 2^{N-K} - 1$$

$$\sum_{i=1}^{t} \binom{N}{i} = \binom{7}{1} = 7 \text{ and}$$

$$2^{N-K} - 1 = 2^{7-4} - 1 = 7$$

Since both sides of the sphere packing bound equation are equal, it can be seen clearly that the Hamming codes are perfect codes and it can correct 1 error.

3.2 Trellis for Linear Block Codes

Let the non zero code word c= (c1... cn) explains the start of c and is denoted as start (c), the smallest integer for i in the condition ci is non zero. Similarly let the non zero code word c= (c1....cn) explains the end of c and is denoted as end (c), the largest integer for i in the condition ci is non zero. Then the span of c or the support interval of c can be defined as the interval [start (c), end (c)] where the span or the support interval of the zero word 0 is an empty interval as [ ]. The span length of c can be defined by the following equation and it is defined as the cardinality of its span.

$$L(c) = \text{end}(c) - \text{start}(c) + 1$$

Where

$$L(0) = 0$$

The method proposed by Wolf (Wolf, 1978) needs parity check matrix H = (h1 h2... hn). Here hi where i can be assigned values 1, 2 ...n and is the ith column of the parity check matrix which has got n-k elements for GF (2). Trellis is an easy way to represent the 2k code words and it has got n+1 set of nodes and each set has got 2n-k nodes. Now in order for the ease of explanation let us consider i as the sets where i= 0, 1... n. The nodes in any set will also consist of another parameter j where j= 0, 1 ... 2n-k-1 and so we can say that the jth node in the ith set has got an index which is expressed as (j, i). The nodes are connected with branches in a certain manner and also uniquely defined by H, we can say that a trellis is formed. The steps for the procedure are explained below.

- For the set i=0, the branches originate only from the node (0, 0) and there will be two branches. One branch with weight 0 and the other with weight 1. The branch which has got the weight 0 will enter the node (0, 1). And the branch with weight which is equal to 1 will only enter the node (b, 1). ‘b’ is

the transpose of the vector h1 is actually the decimal equivalent of the binary number.

- For any other node (j, i) where $i > 0$, which has got incoming branches and also branches are with weight 0 and 1. The destination nodes are determined using the steps shown below.
  - Calculate x, in which x is the binary equivalent of the decimal number represented by j which is mentioned above.
  - Now calculate the binary number $y = t_{i+1} \oplus x$. Here t_{i+1} is a binary number which is shown as the transpose of the vector h_{i+1}.
  - Now consider z as the decimal equivalent of y.
  - For the branch with weight 0 the destination node in set i+1 is node (j, i+1)
  - And for the branch with weight 1 the destination node in se i+1 is node (z, i+1).
- Now repeat the second step again and again for i = 1, 2 ... n-1. By following this procedure a trellis with more paths than the code words will be generated. Now remove all the paths (known as expurgation) which do not end in node (0, n). Thus the remaining will be the 2k unique paths which indicate all the code words in the block code (Buttner et al., 1998).

3.3 Viterbi decoding using trellis

We already know that each branch of a trellis represents a bit in the valid code word. So we can say that the most likely path can be found out by comparing each of the incoming bit or a sample of the received vector (which is called as hard decision and soft decision respectively) with the branch weights. Assume that there are n received symbols for a code word and also they are statistically independent. Therefore the probability of the received sample/bit when compared with the branch weight which is called as a metric can be explained as shown below.

$$Z(y_i, w(x,i),(z,i+1)) = \log(p(y_i|w(x,i),(z,i+1)))$$

Where, w(x,i),(z,i+1) = weight of the branch from the node (x, i) to (z, i+1) and

y_i = ith sample/ bit which is received. When we consider the hard decision implementation the probability of making an error is as shown below.

$$Z(y_i, w(x,i),(z,i+1)) = \begin{bmatrix} \log(1-p) & y_i = w(x,i),(z,i+1) \\ \log(p) & y_i \neq w(x,i),(z,i+1) \end{bmatrix}$$

When input samples/bits are being received corresponding cumulative metrics are being calculated which indicates the most favourable paths. The following rules are been used when Viterbi algorithm is applied to the trellis for decoding.

- Assign zero to the cumulative metric CM00 at node (0, 0).
- For every node in set i+1 which has got atleast one incoming branch we have to calculate one or more metrics. The following computations are to be done depending on the number of branches entering.
  - M0 = CMji + Z (yi, 0)

Where CMji is the cumulative metric at node (j, i) and this metric has to be calculated only if a 0-weight branch enters the node.

- M1 = CMji + Z (yi, 1)
  Where CMji is the cumulative metric at node (j, i) and this metric has to be calculated only if a 1-weight branch enters the node.

- For the node (j, i+1) the cumulative metric at that node is assigned to CMj (i+1) = min (M0, M1). When we take the case of two identical metrics one of them is chosen randomly as the survivor. When only one metric is calculated for a particular node then the cumulative metrics will assume that metrics value. This means in this case CMj (i+1) = M0 or CMj (i+1) = M1. When two branches enter a node in set i+1 one of them will be removed. The one that is most likely to be removed is the branch that has the larger metrics.
- Repeat the above steps for i= 0, 1... n-1 times. When this is done we should get only one path which starts from node (0, 0) and ends in (0, n). Thus the most likely code word can be found out by noting the weights of the branches in the path obtained by following the above steps (Buttner et al., 1998).

4 Result

Let us consider the G matrix: $G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

For this G matrix, the trellis diagram after encoding is shown below:

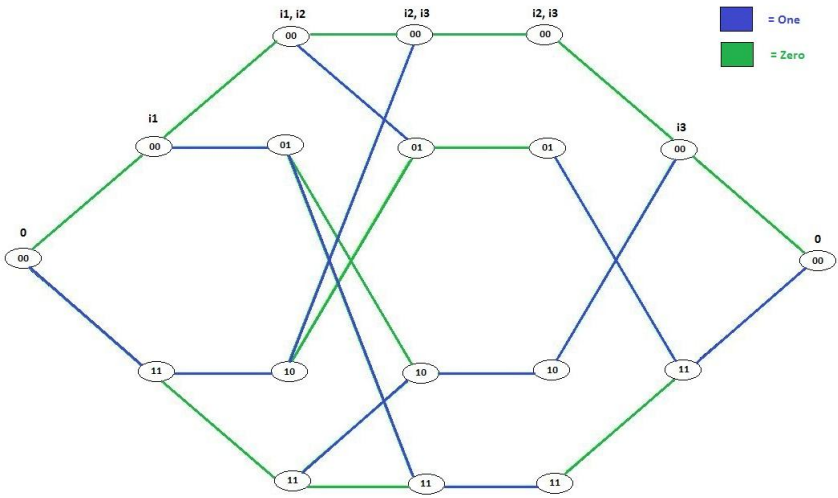

Figure 3: Trellis Diagram

4.1 Hard Decision Decoding

The decoding process for each codeword is done as explained in the previous chapter. The results for hard decision decoding are collated in the tables below.

Received Codeword without errors																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0
2	0	1	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0
3	0	0	1	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1

Table 1: Received codewords without errors for hard decision decoding

Received Codeword with 1 bit error																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	1	1	**0**	0	0	0	1	1	1	0	0	0
2	1	1	1	0	0	0	1	**0**	1	0	0	0	1	1	1	0	0	0
3	1	1	1	0	0	0	1	1	1	0	**1**	0	1	1	1	0	0	0
4	1	1	1	0	0	0	**0**	1	1	0	0	0	1	1	1	0	0	0
5	0	1	0	1	1	0	0	**0**	0	1	1	0	0	1	0	1	1	0
6	0	1	0	1	1	0	0	1	0	**0**	1	0	0	1	0	1	1	0
7	0	1	0	1	1	0	0	1	0	1	1	**1**	0	1	0	1	1	0
8	0	1	0	1	1	0	0	1	0	1	**0**	0	0	1	0	1	1	0
9	0	0	1	0	1	1	0	0	1	**1**	1	1	0	0	1	0	1	1
10	0	0	1	0	1	1	0	**1**	1	0	1	1	0	0	1	0	1	1
11	0	0	1	0	1	1	**1**	0	1	0	1	1	0	0	1	0	1	1
12	0	0	1	0	1	1	0	0	**0**	0	1	1	0	0	1	0	1	1

Table 2: Received codewords with 1 error for hard decision decoding

Received Codeword with 2 bit errors																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	1	1	**0**	**1**	0	0	1	1	1	0	0	0
2	1	1	1	0	0	0	1	**0**	**0**	0	0	0	0	0	0	0	0	0
3	1	1	1	0	0	0	1	1	1	0	**1**	**1**	1	1	0	0	1	1
4	0	1	0	1	1	0	0	**0**	**1**	1	1	0	1	0	1	1	1	0
5	0	1	0	1	1	0	0	1	0	**0**	**0**	0	0	0	0	0	0	0
6	0	1	0	1	1	0	**1**	1	0	1	1	**1**	1	1	0	0	1	1
7	0	0	1	0	1	1	**1**	0	1	**1**	1	1	1	0	1	1	1	0
8	0	0	1	0	1	1	0	**1**	1	**1**	1	1	0	1	1	1	0	1
9	0	0	1	0	1	1	**1**	0	**0**	0	1	1	1	1	0	0	1	1

Table 3: Received codewords with 2 errors for hard decision decoding

From the above results, it is clear that the decoder implemented in the MATLAB software works perfectly for codewords received without any error and for codewords which have 1 bit error. It cannot correct codewords with errors in two bit locations. As explained, it proves that the Hamming code corrects 1 bit error in the received codeword.

4.2 Soft Decision Decoding

Now, for the same G matrix, let us verify the results using soft decision decoding for the same set of codewords.

Received Codeword without errors																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	0.528	0.765	0.664	-0.26	-1.19	-0.76	1	1	1	0	0	0
2	0	1	0	1	1	0	-1.06	1.281	-1.24	0.557	0.55	-0.85	0	1	0	1	1	0
3	0	0	1	0	1	1	-1.06	-1.06	1.449	-0.91	1.063	1.502	0	0	1	0	1	1

Table 4: Received codewords without errors for soft decision decoding

Received Codeword with 1 bit error																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	0.746	1.22	**-0.74**	-1.08	-0.93	-1.37	1	1	1	0	0	0
2	1	1	1	0	0	0	1.304	**-0.96**	1.454	-1.62	-1.06	-1.38	1	1	1	0	0	0
3	1	1	1	0	0	0	0.974	0.389	0.861	-1.57	**1.266**	-1.28	1	1	1	0	0	0
4	1	1	1	0	0	0	**-0.46**	0.939	0.324	-1.27	-0.57	-1.34	1	1	1	0	0	0
5	0	1	0	1	1	0	-1.5	**-0.84**	-0.91	1.011	0.578	-0.64	0	1	0	1	1	0
6	0	1	0	1	1	0	-0.89	0.905	-0.99	**-1.08**	0.447	-1.09	0	1	0	1	1	0
7	0	1	0	1	1	0	-1.26	0.69	-1.37	0.831	0.367	**1.305**	0	1	0	1	1	0
8	0	1	0	1	1	0	-0.84	0.994	-1.01	0.748	**-0.68**	-1.04	0	1	0	1	1	0
9	0	0	1	0	1	1	-1.23	-0.57	0.929	**0.814**	0.907	0.732	0	0	1	0	1	1
10	0	0	1	0	1	1	-1.43	**1.144**	0.732	-1.11	1.175	1.329	0	0	1	0	1	1
11	0	0	1	0	1	1	**0.904**	-0.99	1.016	-0.74	1.483	1.148	0	0	1	0	1	1
12	0	0	1	0	1	1	-1.07	-0.8	**-0.94**	-1.33	1.3	1.097	0	0	1	0	1	1

Table 5: Received codewords with 1 error for soft decision decoding

Received Codeword with 2 bit errors																		
No.	**Transmitted Codeword**						**Received Codeword**						**Decoded Codeword**					
1	1	1	1	0	0	0	1.043	1.163	**-0.92**	**0.702**	-1.05	-1.05	1	1	1	0	0	0
2	1	1	1	0	0	0	0.939	**-1.09**	**-0.52**	-1.08	-1.34	-0.49	0	0	0	0	0	0
3	1	1	1	0	0	0	1.39	0.927	0.524	-1.14	**0.951**	**1.087**	1	1	0	0	1	1
4	0	1	0	1	1	0	-1.16	**-1.1**	**1.004**	0.042	0.855	-0.61	0	0	1	0	1	1
5	0	1	0	1	1	0	-1.34	1.295	-0.89	**-1.01**	**-0.94**	-1.49	0	0	0	0	0	0
6	0	1	0	1	1	0	**0.973**	1.507	-0.97	1.013	0.768	**0.99**	1	1	0	0	1	1
7	0	0	1	0	1	1	**1.073**	-0.87	0.882	**0.925**	1.64	0.286	1	0	1	1	1	0
8	0	0	1	0	1	1	-0.29	**1.107**	1.316	**0.474**	0.813	0.912	0	1	1	1	0	1
9	0	0	1	0	1	1	**1.103**	-0.66	**-0.68**	-1.21	1.081	0.701	1	1	0	0	1	1

Table 62: Received codewords with 2 errors for soft decision decoding

5 Conclusion

This project involves in the brief study of error correction coding. Also, a detailed study of convolutional coding and block codes has been covered with more emphasis on linear block codes. A software implementation of the encoding and decoding of the shortened (7, 4) Hamming code has been completed in MATLAB. The code has been tested with various input codeword inputs to the decoder and the results have been summarized in the previous chapter. The software implementation includes both the soft decision decoding and hard decision decoding of the receiver output and the Viterbi decoding algorithm is applied to get the output of the decoder.

6 Reference

Buttner, W. H., Staphorst, L. & Linde, L. P. (1998) Trellis decoding of linear block codes. *In:* Communications and Signal Processing, 1998. COMSIG '98. Proceedings of the 1998 South African Symposium on, 1998. 171-174.

Lin, S. 1970. *An Introduction to Error-Correcting Codes*, New Jersey, Prentice Hall.

Lin, S. & Costello, D. J. 2004. *Error Control Coding, Second Edition*, Prentice-Hall, Inc.

Michelson, A. M. & Levesque, A. H. 1985. *Error control techniques for digital communication*, Wiley – Interscience.

Proakis, J. 2000. *Digital Communications*, McGraw-Hill International.

Wade, G. 2000. *Coding Techniques: An Introduction to Compression and Error Control*, Palgrave Macmillan.

Wolf, J. 1978. Efficient maximum likelihood decoding of linear block codes using a trellis. *Information Theory, IEEE Transactions on*, 24**,** 76-80.