CHAPTER 15
Channel Coding

15.1 Repetition Code:

In communication we are concerned with both speed and accuracy. Efficiency is a measure of redundancy. High efficiency entails zero redundancy. But when we consider transmission in a noisy channel, we also have to consider accuracy. Redundancy is added in a controlled manner to protect against errors, to withstand the effects of various channel impairments such as noise, interference and fading. For example, in a system with two input symbols 0,1 we may use redundancy such that 0 is represented by 000 and 1 by 111. Thus redundancy is 2/3 and the information code rate r is 1/3 bit per digit. Using the majority rule, we can identify 001 or 010 or 100 as zero and 110, 011, 101 as1. It is important to find the relation between the probability of error and the number of repetitions. The binomial distribution provides the answer. It refers to a process with two outcomes p and $\bar{p}$ stating that the probability $p(r_e)$ of getting r_e errors in n trial is given by the $(r+1)^{th}$ term of the binomial expansion of $\left(p+\bar{p}\right)^n$, which is

$$p(r_e)=\frac{n\,!}{(n-r_e)!\,r_e\,!}\,p^{re}(\bar{p})^{n-r_e} \tag{15 – 1}$$

When we use 3 digits we can tolerate an error and still get a correct symbol.

Ex. 15.1

Assume 5 digits codeword to represent a binary symbol in a binary symmetric channel with binary probability of error $p=0.01$, find $p(e)$?

Solution

Using the majority rule. A symbol is interpreted as 0 if it has more 0's than 1's and vice versa. Two errors can be tolerated without producing a symbol error. The symbol error $p(e)$ is given by the sum of probabilities of 3, 4, 5 errors in five trials.

From eqn. (15-1)

$$(\bar{p}+p)^5=\bar{p}^5+5\bar{p}^4\,p+10\bar{p}^3\,p^2+10\bar{p}^2\,p^3+5\bar{p}^4\,p+p^5 \tag{15 – 2}$$

$$p(r)\;:p(0)\quad p(1)\quad p(2)\qquad p(3)\qquad p(4)\quad p(5)$$

$$p(e)=p(3)+p(4)+p(5)=9.8\times10^{-6}$$

We see that the code rate 1/5 bits / binary digit. Thus, we can decrease $p(e)$ at the expense of deceasing r.

15.2 Hamming Distance:

We have seen that a repetition code reduces the data rate dramatically. On the other extreme we may select a codeword out of a selection of codewords to represent a symbol. Using 5 digits codewords would produce 32 possible codewords. If we forget about redundancy, we could have used all of the 32 codewords to represent different symbol such as. $A = 00000$, $B = 00001, C = 00010$ and so on. In this case a single binary error leads to a symbol error, i.e.

$$p(e) = p(1) + p(2) + p(3) + p(4) + p(5)$$
$$= 1 - p(0) = 0.049$$

The code rate r is 1bit / binary digit. A comparison is made between these two extremes by selecting a number between 2 and 32. Consider the selection of 4 symbols as follows

Symbol		**state representation**
$A =$	0 0 0 0 0	00
$B =$	0 0 1 1 1	01
$C =$	1 1 0 0 1	10
$D =$	1 1 1 1 0	11

There are many other possible choices. But we notice here that every codeword differs from all others by at least 3 digit positions. The number of digit positions in which two codewords differ is called the Hamming distance and the least difference between any members of a given set of codewords is called the minimum Hamming distance $d_{\min}$. It is clear that single errors can be tolerated since received codewords with one error will still be closer to the transmitted codewords than to any other member of the set. However, a symbol error will occur for 2 or more errors (according to majority rule). Thus,

$$p(e) = p(2) + p(3) + p(4) + p(5)$$
$$\sim p(2) = 9.7\times10^{-4}$$

The code rate $r = 2/5 = 0.4$ bits / binary digit. Thus, we have in Table (15.1)

Table (15.1) Effect of r on $p(e)$ for 5 digit codeword

Number of symbols	**Number of selections**	$P(e)$	r
2^1	2 (binary)	9.8×10^{-6}	0.2
2^2	4	9.7×10^{-4}	0.4
2^5	32	0.049	1

Thus, increasing the code rate r is at the expense of increasing probability of symbol error. From this example we see that a comparison between low data rate and high data rate may be possible by using long groups of digits or codewords which are sufficiently different from one another to make $p(e)$ acceptably low. This is the essence of Shannon's second theorem, which may be stated in a relevant variation as if coding is in long groups of n binary digits then provided that the number of selected groups $M \leq 2^{nC}$ where C is the channel capacity, then the symbol error rate $p(e)$ can be made arbitrarily small as $n \to \infty$

$$\lim_{n\to\infty} p(e) \to 0 \text{ provided } M \leq 2^{nC} \qquad (15-3)$$

Assuming that the M selected codewords are equiprobable.

$$r = \frac{\log_2 M}{n} = \frac{k}{n} \quad \text{bits/symbol}$$

$$= \frac{\log_2 2^{nc}}{n} = C \qquad (15-4)$$

Thus provided $r \leq C$, we can make $p(e) \to 0 \, as \, n \to \infty$.

This is a surprising result, namely that, irrespective of the binary error probability p, information can be transmitted at any code rate up to the channel capacity with virtually no net error in the decoded output symbols. Applying this result to the case with $p = 0.01$, we have $C = 1 - H(p) = 0.919 \; bits$

For 2 codewords $\frac{r}{C} = \frac{0.2}{0.919} = 0.22$

For 4 codewords $\frac{r}{C} = \frac{0.4}{0.915} = 0.44$

For 32 codewords $\frac{r}{C} = \frac{1}{0.919} > 1$

In the last case the error is great. Note that $p(e)$ doesn't tend to zero for 2 or 4 codewords since n is not sufficiently large. The error rate falls as n increase to an extent depending on r/C .

15.3 Error Detection and Correction:

Shannon's theorem gives theoretical limits but does not provide codes. It is important to distinguish between error detection and error correction. Suppose we have a set of codewords with $d_{\min} = 2$. A single binary error will produce a word not in the list of those selected, so that the presence of a binary error will be detected but not corrected, i.e., there will be no way of knowing which codeword was actually transmitted. However if $d_{\min} = 3$, a single error will produce a word nearer to the correct word so a single error can be corrected. In general to detect t digit errors

$$d_{\min} \geq t + 1 \tag{15 – 5}$$

But to correct t digit errors

$$d_{\min} \geq 2t + 1 \tag{15 – 6}$$

In other words, it is required that the code be constructed such that when error occurs in an allowable code sequence, a nonallowable codeword is produced. But if the error occurs such that another allowable codeword is produced then such an error is unrecoverable.

Errors may be divided to random or burst. A random error is such that there is no correlation between bit positions in error. While in an error, bits in burst error are contained in a whole sequence as in the case of electromagnetic impulses.

For example 1ms impulse could corrupt 10 bits in a sequence of a digital transmission at 9600 b/s. Binary error rate (BER) of 10^{-5} is usually accepted for public switched telephone network (PSTN). To counter the effects of this noise we need error control techniques involving error detection and if possible forward error correction.

There are two types of channel coding: block coding and convolutional coding. In a block code, a channel coder (n,k) takes each set of k incoming digits and maps it into a set of n outgoing digits, where $n > k$,where $r = k/n$ is the code rate and $(n-k)1k$ is redundancy. The extra $n-k$ digits (redundancy digits or check digits) introduced in the stream by the channel coder are added so that we can detect transmission errors and or if possible remove them at the receiver. The channel decoder performs the reverse operation. It maps each set of incoming n digits into the original k digits using the $(n-k)$ digits to detect and possibly correct any errors.

Thus, in a block code, data blocks are mapped into code blocks in such a way that a change is any code block due to noise creates a set of symbol not allowed by the code. Hence, the nonallowable block can be detected and possibly corrected.

The simplest way after an error is detected is to request for retransmission. In case forward error correction (FEC) is available, the digits in error must be inverted. An efficient code uses long blocks with a small portion of extra digits for error detection and correction. However, the longer the block length the more information is lost if the code breaks down. It also takes longer time to encode and decode so extra delays are added to the system. So for real time usage, the block length must be a compromise. The simplest block code is a single parity check bit code. Each set of k incoming digits is mapped to $(k+1)$ outgoing bits. The additional digit is included to ensure if we add all digits together we get a total of 0, using modulo 2 addition, which is $(0+0=0,\ 1+0=1,\ 0+1=1,\ 1+1=0)$. Channel coding where one digit is added to create a total sum of 0 is called even parity, whereas if the sum is 1, it is called odd parity.

For example for even parity, take an input to the channel coder as 101, the output is 1010. The parity check digit is 0, to indicate that the sum is 0. The decoder checks on the sum. If it is 0, it recognizes that no error is detected. If it is 1 then it recognizes that an error has occurred. But if two errors occur the single parity check coder will not recognize either. It can detect only an odd number of errors but will not detect the number of such errors. The single parity check coder naturally will not be able to locate the digit in error let alone correct it

The concept of parity can be extended to cover a larger number of words of data by storing them in an array of rows and columns. This is called rectangular a product code. Let us assume a set of 9 digits configured in a 3×3 matrix.

A parity digit is created for each row and for each column. With the addition of this digit to each row the total sum of each row is now 0. With the addition of an extra digit to each column the total modulo 2 sum of each column is 0. These digits now in a 4×4 matrix are sent serially across the channel as a set of 16 digits (Fig. 15.1)

The channel decoder returns the incoming serial digits back to a matrix form (Fig.15.2) noting that one of the received bits is in error. The channel decoder computes the sum for each column and the sum of each row. Since there is an error in one of the columns, then the sum is 1 for that column. Since there is also an error in one of the rows, then the sum is 1 for that row. Thus, we can locate the exact column and exact row indicating when the single error has occurred. Thus, we can revere its value. Thus, a rectangular coder can easily correct one digit error.

Ex. 15.2

Consider the following Hamming code. Parity digits C_0, C_1, C_2, C_3 are interspersed in the code (Table 15.2).

Table (15.2) Ex. (15.2)

	Position	Digits checked
C_0	1	1,3,5,7
C_1	2	2,3,6,7
C_2	4	4,5,6,7
C_3	8	

Verify the coder and decoder operation.

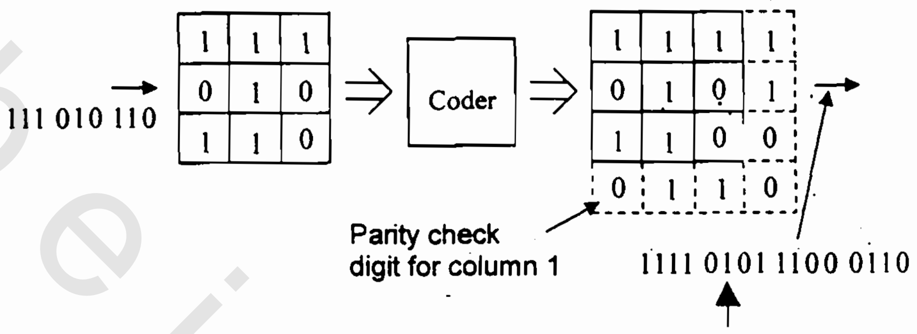


Fig. (15.1) Rectangular coder

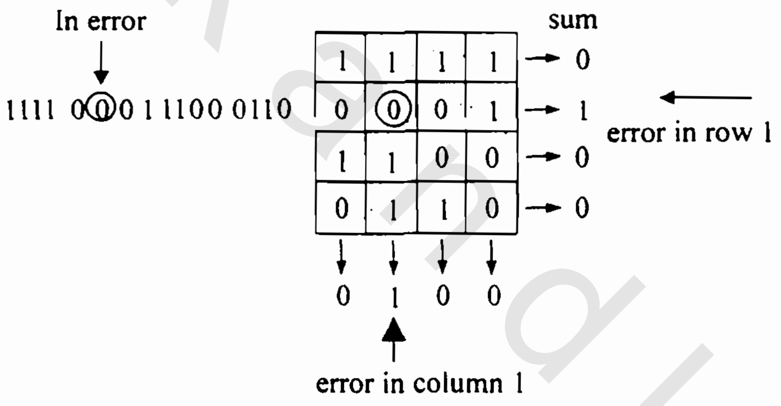


Fig. (15.2) Rectangular decoder

Solution

Ignoring for the moment column 8 (for double error detection). For 16 symbols we require 4 data digits: 3, 5, 6, 7. Three check digits are needed in positions 1, 2, 4. This code is denoted (7, 4) since $n=7$, $k=4$. Any suitable code can be used. As an example we have Table (15.3).

The binary code (data) is first written out in digit positions 3, 5, 6, and 7 leaving positions 1, 2, and 4 blank C_0 is then placed in position 1, being a parity check on digit positions 1, 3, 5 and 7. The process is repeated for C_1 and C_2 using the combinations in table 15.2 ignoring the digit in position 8 for now. Suppose $D=1000011$ was transmitted and received an 1010011, i.e. with the third digit in error.

Table 15.3 Hamming code (7,4)

		(C_0)	(C_1)		(C_2)				(C_3) (P)
		1	2	3	4	5	6	7	8
1)	A	0	0	0	0	0	0	0	0
2)	B	1	1	0	1	0	0	1	0
3)	C	0	1	0	1	0	1	0	1
4)	D	1	0	0	0	0	1	1	1
.									
.									
15)	O	0	0	0	0	1	1	0	1
16)	P	1	1	1	1	1	1	1	1

The parity checks are carried out producing the binary checking number $C_2\, C_1\, C_0 = 011 = 3_{10}$. Thus, the third digit is in error and is therefore complemented and D is decoded. If no errors occur, the checking number will be zero. But if two or more errors occur, it will not provide meaningful results. The code can be made to detect double errors if an extra overall parity digit is added. This is placed in position 8 producing (8,4) code. This extra digit is outside the Hamming process. Note that since $(C_2C_1C_0)$ is required to give the error position, C_0 must be 1 if the error is in positions 1, 3, 5, 7, since the binary equivalent of these numbers has a "1" in the least significant position, similarly for C_1 and C_2 (Prob. 15-1).

15.4 Linear Block Codes:

To generate an (n,k) block code, the channel encoder accepts information in successive k digits blocks. For each block, it adds $n-k$ redundant digits that are algebraically related to the k message digits, producing an encoded block of n digits or a codeword. The channel encoder produces digits at the rate of $R_c = (n/k)R_s$ where R_s is the bit rate of information generated by the source, $r = k/n$ is the code rate $o \le r \le 1$, *and* R_c is the channel data rate digits/s. Consider an (n,k) block code in which the first k digits is always the message. Let $m_0, m_1 .. m_{k-1}$ be the message bits, we have 2^k distinct message blocks. Let this sequence of message bits be applied to a linear block encoder producing an n bit codeword $x_0, x, .. x_{n.1}$. Let $b_0, b_1 .. \quad b_{n-k-1}$ denote the binary digits in the codeword (Fig. 15.3).

Thus,

$$x_i = \begin{matrix} b_i & 0,1..n-k-1 \\ m_{i+k-n} & n-k, n-k+1...n-1 \end{matrix} \qquad (15-7)$$

The $n-k$ parity bits are linear sums of the message bits

$$b_i = p_{i0}\ m_0 + p_{i1}\ m_1 + ...\ p_{i1\,k-1}\ m_{k-1} \qquad (15-8)$$

$$p_{ij} = \begin{matrix} 1 & \textit{if } b_i \text{ depends on } m_j \\ 0 & 0 \quad \textit{otherwise} \end{matrix}$$

We call this code systematic linear code. These coefficients are chosen so that we have $(n-k)$ linearly independent equations, i.e., no equation in the set may be expressed as a linear combination of the remaining ones. Eqns. (15-7) and (15-8) define the mathematical structure of the (n,k) linear block code. In matrix notation we have $1\times k$ message row vector **m** and $1x(n-k)$ parity row vector $\boldsymbol{b}$ and $1\times n$ code vector $\boldsymbol{X}$ so

$$\boldsymbol{m} = [m_0\ , m_1\ ..\ m_{k-1}] \qquad (15-9)$$

$$\boldsymbol{b} = [b_0\ , b_1\ ..\ b_{n-k-1}] \qquad (15-10)$$

A set of simultaneous equations may be written in matrix notation as

$$\boldsymbol{b} = \boldsymbol{m}\ \boldsymbol{P} \qquad (15-11)$$

where $\boldsymbol{P}$ is the $k\times(n-k)$ coefficient matrix

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{10} & .. & p_{n-k-1,0} \\ p_{01} & p_{11} & .. & p_{n-k-1,1} \\ \vdots & \vdots & & \vdots \\ p_{0,k-1} & p_{1,k-1} & .. & p_{n-k-1,k-1} \end{bmatrix} \qquad (15-12)$$

From eqns (15-9), (15-10), we may write X as a partitioned row vector as

$$\boldsymbol{X} = [\boldsymbol{b} \vdots \boldsymbol{m}] \qquad (15-13)$$

Substituting eqn. (15-11) and factoring out $\boldsymbol{m}$,

$$\boldsymbol{X} = \boldsymbol{m}[\boldsymbol{P} \vdots \boldsymbol{I}_k] \qquad (15-14)$$

where $\boldsymbol{I}_k$ is $k\times k$ identity matrix

$$\boldsymbol{I}_k = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & 1 \end{bmatrix} \qquad (15-15)$$

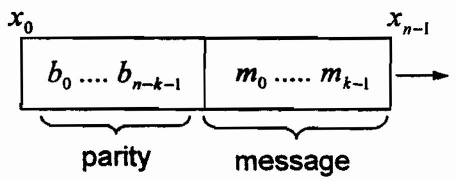


Fig. (15.3) Structure of the codeword

We define the $k \times n$ generator matrix as

$$G = [P \vdots I_k] \tag{15-16}$$

so that eqn. (15.14) can be rewritten as

$$X = m\,G \tag{15-17}$$

The generator matrix G has its k rows linear independent., i.e., it is not possible to express any row of the matrix G as a linear combination of the other rows. The full set of codewords is generated from eqn. (15-17) by letting m range through the set of all 2^k binary k tuples $(1 \times k \text{ vectors})$. We note that the sum of any two codewords is another codeword. This basic property of linear block codes is called closure. To verify it consider two codewords X_i and X_j corresponding to two message m_i and m_j, respectively. Using eqn. (15-17)

$$X_i + X_j = m_i\,G + m_j\,G$$

$$= (m_i + m_j)\,G \tag{15-18}$$

The sum of any two message gives a new message vector, hence the sum of two code vectors gives another code vector in the code.

A second basic property for a linear code is that it contains an all 0 codeword.

Let us define the $(n-k)xn$ parity check matrix H as

$$H = [I_{n-k} \vdots P^T] \tag{15-19}$$

Where P^T is an $(n-k) \times n$ matrix representing the transpose of the coefficient matrix P and I_{n-k} is the $(n-k) \times (n-k)$ identity matrix. Thus,

$$H\,G^T = [I_{n-k} \vdots P^T] \begin{bmatrix} P^T \\ \cdots \\ I_k \end{bmatrix} \tag{15-20}$$

$$= P^T + P^T \tag{15-21}$$

Since in modulo 2 arithmetic $\boldsymbol{P}^T + \boldsymbol{P}^T = \boldsymbol{0}$,

$$\boldsymbol{H}\ \boldsymbol{G}^T = \boldsymbol{0} \qquad (15-22)$$

Post multiplying eqn. (15-17) by $\boldsymbol{H}^T$, then

$$\boldsymbol{X}\ \boldsymbol{H}^T = \boldsymbol{m}\boldsymbol{G}\ \boldsymbol{H}^T = \boldsymbol{0} \qquad (15-23)$$

Ex. 15.3

Verify that the repetition code is a linear block code.

Solution

A single message bit is encoded into a block of identical n bits producing an $(n,1)$ block. There are only two codewords is the code, an all zero code and an all 1 codeword. Consider $k=1,\ \ n=5$. We have 4 parity bits that are the same as the message bit.

$$\boldsymbol{G} = [1\ 1\ 1\ 1 \vdots 1]$$

$$\boldsymbol{P} = [1\ 1\ 1\ 1]$$

$$\boldsymbol{P}^T = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & \vdots & 1 \\ 0 & 1 & 0 & 0 & \vdots & 1 \\ 0 & 0 & 1 & 0 & \vdots & 1 \\ 0 & 0 & 0 & 1 & \vdots & 1 \end{bmatrix}$$

Note that $\boldsymbol{H}\boldsymbol{G}^T = \boldsymbol{0}$

$$\boldsymbol{H}\ \boldsymbol{G}^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = 0$$

Ex. 15.4

Examine the following coding assignment that describes a (6,3) code.

m	*X*	
000	000	000
100	110	100
010	011	010
110	101	110
001	101	001
101	011	101
011	110	011
111	000	111

Verity that X constitutes a linear code, find P , G , and H

Solution

From eqn: (15.11), $b = m\,P$, for $k = 3$, $n = 6$

$$\begin{pmatrix} 000 \\ 110 \\ 011 \\ 101 \\ 101 \\ 010 \\ 110 \\ 000 \end{pmatrix} = \begin{pmatrix} 000 \\ 100 \\ 010 \\ 110 \\ 001 \\ 101 \\ 011 \\ 111 \end{pmatrix} \begin{pmatrix} P_{00} & P_{10} & P_{20} \\ P_{01} & P_{11} & P_{21} \\ P_{02} & P_{12} & P_{22} \end{pmatrix}$$

$$\begin{pmatrix} 000 \\ 110 \\ 011 \\ 101 \\ 101 \\ 010 \\ 110 \\ 000 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ p_{00} & p_{10} & p_{20} \\ p_{01} & p_{11} & p_{21} \\ p_{00}+p_{01} & p_{10}+p_{11} & p_{20}+p_{21} \\ p_{02} & p_{12} & p_{22} \\ p_{00}+p_{02} & p_{10}+p_{12} & p_{20}+p_{22} \\ p_{01}+p_{02} & p_{11}+p_{12} & p_{21}+p_{22} \\ p_{00}+p_{01}+p_{02} & p_{10}+p_{11}+p_{12} & p_{20}+p_{21}+p_{22} \end{pmatrix}$$

$$p_{00}=1 \ , \ p_{10}=1 \ , \ p_{20}=0$$
$$p_{01}=0 \ , \ p_{11}=1 \ , \ p_{21}=1$$
$$p_{00}+p_{01}=1 \ , \ p_{10}+p_{11}=0 \ , \ p_{20}+p_{21}=1$$
$$p_{02}=1 \ , \ p_{12}=0 \ , \ p_{22}=1$$
$$p_{00}+p_{02}=0 \ , \ p_{10}+p_{12}=1 \ , \ p_{20}+p_{22}=1$$
$$p_{01}+p_{02}=1 \ , \ p_{11}+p_{12}=1 \ , \ p_{21}+p_{22}=0$$
$$p_{00}+p_{01}+p_{02}=0 \ , \ p_{10}+p_{11}+p_{12}=0 \ , \ p_{20}+p_{21}+p_{22}=0$$

From this we find

$$\boldsymbol{P}=\begin{bmatrix}1&1&0\\0&1&1\\1&0&1\end{bmatrix}$$

and

$$\boldsymbol{G}=\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&1&0&0&1\end{bmatrix}$$

$$\boldsymbol{P}^T=\begin{bmatrix}1&0&1\\1&1&0\\0&1&1\end{bmatrix}$$

From eqn. (15 – 19)

$$\boldsymbol{H}=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&1&1&0\\0&0&1&0&1&1\end{bmatrix}$$

We can verify $\boldsymbol{H}\,\boldsymbol{G}^T=\boldsymbol{0}$ and $\boldsymbol{X}\boldsymbol{H}^T=\boldsymbol{0}$

15.5 Syndrome Ddecoding:

The generator matrix $\boldsymbol{G}$ is used in the encoding process at the transmitter. On the other band, the parity check matrix $\boldsymbol{H}$ is used in the decoding process at the receiver. Let $\boldsymbol{Y}$ be the (1×1) receiver vector of a signal sent over a noisy channel.

$$\boldsymbol{Y}=\boldsymbol{X}+\boldsymbol{e} \qquad (15-24)$$

where the vector $\boldsymbol{e}$ is the error vector. The i^{th} element of $\boldsymbol{e}$ is zero if the corresponding elements are the same in both $\boldsymbol{X}$ and $\boldsymbol{Y}$, while it equals 1 if there is an error. The receiver has the task of decoding the code vector X from the received vector Y. We define a $1\times(n-k)$ vector called the syndrome **S** as

$$S = YH^T \qquad (15-25)$$

It has the following properties

1) We note first that the syndrome depends only on the error pattern and not on the transmitted codeword.

From eqns. (15-24), (15-23) and (15-25)

$$S = (X+e)H^T = X H^T + e H^T$$
$$= e H^T \qquad (15-26)$$

Hence, the parity check matrix H of a code permits us to computer S, and hence, the error pattern.

2) All error patterns that differ at most by a codeword have the same syndrome.

We know that for a k bit message there are 2^k distinct code vectors X_i ,$i = 0.1..(2^k-1)$

For any error vector e we define 2^k distinct vectors e_i as

$$e_i = e + X_i \qquad i = 0 , 1 .. 2^{k-1} \qquad (15-27)$$

The set of vectors e_i ,$i = 0,...(2^k-1)$ is called a coset of the code. The coset has 2^k elements that differ at most by a code vector. There are 2^{n-k} possible cosets for an (n,k) linear block code. Multiplying both sides of eqn. (15-27) by the matrix H

$$e_i\ H^T = e\ H^T + X_i\ H^T \qquad (15-28)$$

Using eq: (15.23),

$$e_i\ H^T = e\ H^T \qquad (15-29)$$

Thus, each coset is characterized by a unique syndrome. In other words, S provides some information about the error pattern e but not enough for the decoder to determine uniquely the exact transmitted vector. Even so, it reduces the search from 2^n to 2^k, which constitutes the coset corresponding to S.

3) The syndrome S is the sum of the columns in the matrix H corresponding to the error location. To show this, let H be expanded is terms of columns

$$H = [h_1 , h_2 , ... h_n] \qquad (15-30)$$

Substituting in eqn. (15-26),

$$S = e\,H^T = [e_1\ e_2\ \dots e_n]\begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \\ h_2^T \end{bmatrix}$$

$$S = \sum_{1=1}^{n} e_i\ h_i^T \qquad (15-31)$$

When h_i^T is the i^{th} row of the matrix H^T and e_i is the i^{th} element of the error pattern e if H^T is $\begin{bmatrix} h_{11}^T & h_{21}^T & h_{31}^T \\ h_{12}^T & h_{22}^T & h_{32}^T \\ h_{13}^T & h_{23}^T & h_{33}^T \end{bmatrix}$

Thus

$$S = [e_1\ e_2\ e_3]\begin{bmatrix} h_{11}^T & h_{21}^T & h_{31}^T \\ h_{12}^T & h_{22}^T & h_{32}^T \\ h_{13}^T & h_{23}^T & h_{33}^T \end{bmatrix}$$

$$= \left[\left(e_1\ h_{11}^T + e_2\ h_{12}^T + e_3\ h_{13}^T\right)\ \left(e_1\ h_{21}^T + e_2\ h_{22}^T + e_3\ h_{23}^T\right)\ \left(e_1\ h_{31}^T + e_2\ h_{32}^T + e_3\ h_{33}^T\right)\right]$$

$$= \left[e_1\left(h_{11}^T + h_{21}^T + h_{31}^T\right) e_2\left(h_{12}^T + h_{22}^T + h_{32}^T\right) e_3\left(h_{13}^T + h_{23}^T + h_{33}^T\right)\right] \qquad (15-32)$$

Thus, S equals the sum of those rows of the matrix H^T corresponding to the error location in e .

4) With syndrome decoding, an (n,k) linear block code can correct up to t errors per codeword, provided that n and k satisfy the Hamming bound

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i} \qquad (15-33)$$

Where

$$\binom{n}{i} = \frac{n\,!}{(n-i)\,!\ i\,!} \qquad (15-34)$$

To show this, we have a total of 2^{n-k} syndromes. For an n bit codeword there are $\binom{n}{i}$ multiple error patterns, where i is the number of error locations in the $n \times 1$ error pattern e . The total number of all possible error patterns is the sum of $\binom{n}{i}$ for $i = 0,1..\ t$, where t is the maximum number of error locations in e.

Therefore, if an (n,k) linear block code is to be able to correct up to t errors the total number of syndromes must not be less than the total number of possible error patterns, hence eqn. (15-33) must be satisfied.

5) The syndrome S does not uniquely specify the actual error pattern e. Rather, it identifies the coset that the error pattern belongs to. The most likely error pattern within the coset for a certain S is the one with the largest probability, assuming that the channel noise is additive. Hence, the decoding procedure goes as follows.
   a) For the received vector Y, compute the syndrome.
   $$S = Y\ H^T = e\ H^T \qquad (15-35)$$
   b) Within the coset belonging to S choose e_o, the error pattern with the largest probability.
   c) Compute the code vector of expected transmitted code vector $\hat{X}$
   $$\hat{X} = Y + e_0 \qquad (15-36)$$

Note that modulo 2 addition or subtraction equally apply.

6) We have seen form eqns. (15-5) and (15-6) that if we have two codewords and we want the decoder to be able to correct up to t digit errors, then the decoder will be able to detect the error if $d_{\min} \geq t+1$, but will be able to correct the error only if $d_{\min} > 2t+1$. We can visualize this by the diagram shown (Fig. 15.4). It shows two codewords represented by two points x_1, x_2 Around each, is a sphere of radius t representing the number of digits of an error that need be detected or corrected $(t = 1,2..)$ as in Fig. (15.4). In all cases, an error may be detected but cannot be corrected except when $d_{\min} \geq 2t+1$, since only in this case, the codeword received in error may be assigned to one of the allowable codewords.

Thus, we must have
$$t \leq \frac{1}{2}(d_{\min} - 1) \qquad (15-37)$$

Considering a pair of code vectors X and Y, we know that the Hamming distance $d(x,y)$ is the number of locations in which their respective elements differ. We define the Hamming weight $w(x)$ of a code vector X as the number of nonzero elements in the code vector or the distance between the code vector and an all zero code vector. The minimum distance $d_{\min}$ of a linear block code is the smallest Hamming distance between any pair of code vectors in the code

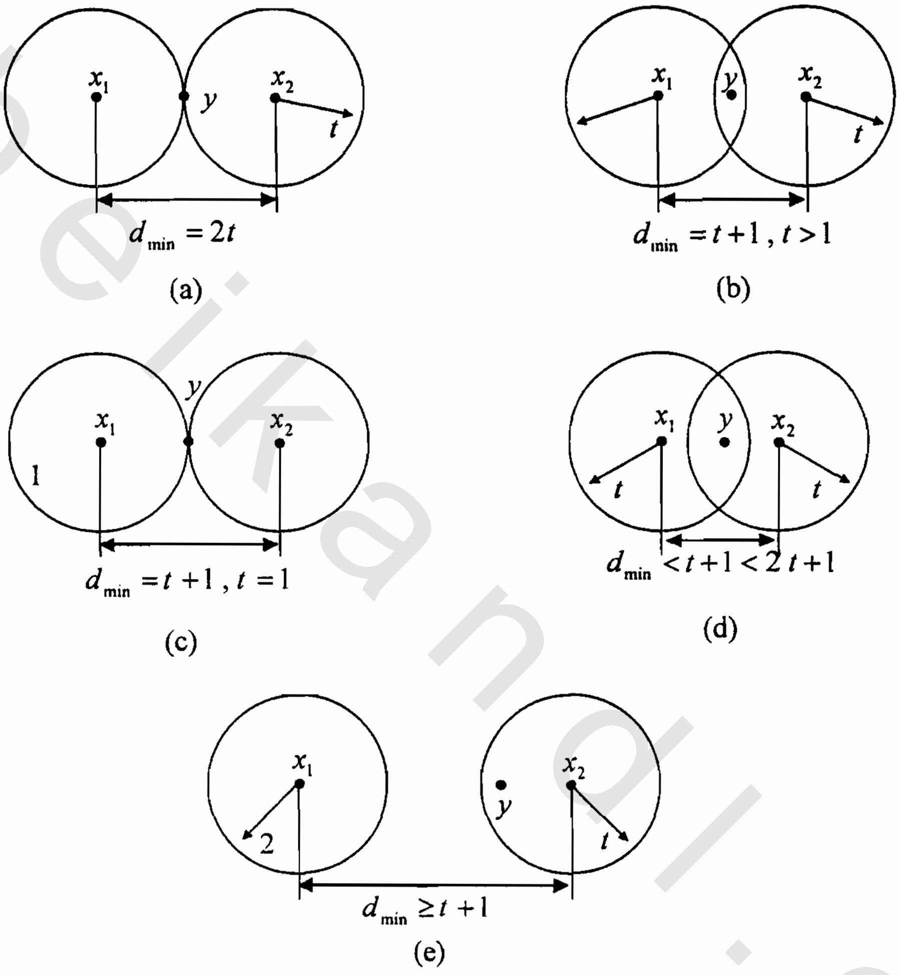


Fig. (15.4) Hamming distance representation

a) $d_{min} = 2t$ b) $d_{min} = t+1, t>1$

c) $d_{min} = t+1, t=1,$ d) $d_{min} < (t+1) < 2t+1$

e) $d_{min} \geq 2t+1$

and consequently, the smallest Hamming weight difference between any pair of the code vectors. From the closure property of linear block codes, the sum or difference of two code vectors is another code vector. Thus, the minimum distance of a linear block code is the smallest Hamming weight of the nonzero codewords in the code.

From eqn. (15-23), a linear block is defined by the set of all code vectors for which $XH^T = 0$. Expressing H as in eqn. (15-30), then for a code vector to satisfy the condition $XH^T = 0$, the vector X must have 1s in such positions that the corresponding rows of H^T sum to O. But the number of 1s in a code vector is the Hamming weight of the code vector or the minimum distance of the code. Thus, the minimum distance of a linear block is equal is to the minimum number of rows of H^T over all codewords that sum to O, i.e.,

$$[x_1, x_2 \dots x_n] \begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \\ h_n^T \end{bmatrix} = 0 \qquad (15-38)$$

We note that no column of H (row of H^T) can be all zeros or error in the corresponding codeword would be undetectable. Also, all columns of H (rows of H^T) must be unique, if two columns of H were identical, errors in the corresponding codeword positions would be indistinguishable. Now, let us see how syndrome decoding is performed. Performing eqn. (15-25), we check if Y is a member of the coset. If S is 0, then Y is a member of the coset. If Y contains detectable errors the syndrome (meaning symptoms of a disease - meaning error) has nonzero value. From eqns. (15-25), (15-26), we see that the syndrome test performed on either the corrupted code vector or the error pattern that caused it yields the same syndrome.

Ex. 15.5

A codeword $X = 101110$ is transmitted, and the vector $Y = 001110$ is received. Find the syndrome vector. Given

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

For the following codewords

massage vector	codewords
000	000000
100	110100
010	011010
110	101110
001	101001
101	011101
011	110011
111	000111

Solution
From eqn. (15.25),

$$S = Y\ H^T$$

$$= [0\,0\,1\,1\,1\,0]\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&0\\0&1&1\\1&0&1\end{bmatrix}$$

$$= [1\,0\,0]$$

From eq: (15.26), noting by comparing X and Y that

$$e = 1\,0\,0\,0\,0\,0$$

$$S = e\ H^T$$

$$= [1\,0\,0\,0\,0\,0]\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&0\\0&1&1\\1&0&1\end{bmatrix} = 1\,0\,0$$

which is the syndrome of the error pattern

Ex. 15.6 (Case Study)

Consider Hamming code (n,k) with properties $n = 2^m - 1$, $k = 2^m - m - 1$, $n - k = m$, $m \geq 3$. For the case $(7,4)$ with the following G matrix

$$G = \begin{bmatrix}1&1&0&\vdots&1&0&0&0\\0&1&1&\vdots&0&1&0&0\\1&1&1&\vdots&0&0&1&0\\1&0&1&\vdots&0&0&0&1\end{bmatrix}$$

$$\qquad\quad \mathbf{P} \qquad\qquad\qquad \mathbf{I}_k$$

Solution

$$\boldsymbol{H} = \left[\boldsymbol{I}_{n-k} \vdots \boldsymbol{P}^T \right]$$

$$= \begin{bmatrix} 1 & 0 & 0 & \vdots & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & \vdots & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & \vdots & 0 & 1 & 1 & 1 \end{bmatrix}$$

With $k = 4$, there are $2^k = 16$ distinct message words listed in Table (15.4).

For a given message word, the corresponding codeword is obtained using eqn. (15-17), resulting in the 16 codewords listed in the table. We have also listed the weights of the individual codewords. Since the smallest of the Hamming weights for the nonzero codeword is 3, it follows that the minimum distance of the code is 3.

To illustrate the relation between $d_{\min}$ and the structure of $\boldsymbol{H}$ matrix, consider the codeword 0110100. Referring to eqn. (15-38), the nonzero elements of this codeword and the second, third and fifth rows of $\boldsymbol{H}^T$ (or columns of $\boldsymbol{H}$) are pointed out yielding 0. Performing similar calculations for the remaining 14 nonzero codewords, we find that the smallest number of columns in $\boldsymbol{H}$ that sums to zero is 3, thus $d_{\min} = 3$

Table (15.5) shows seven distinct single error patterns and the corresponding syndromes. Each single error pattern is associated with a unique syndrome and corresponds to the particular column of $\boldsymbol{H}$ (or row of $\boldsymbol{H}^T$). The zero syndrome signifies no error. Thus, the pattern of the syndrome is the pattern of $\boldsymbol{H}$ column (or $\boldsymbol{H}^T$ row) and the order of such column or row gives the position of the error and hence the error pattern. This error pattern points out directly which codeword is corrupted. Given $\boldsymbol{Y}$ and $\boldsymbol{e}$, we correct $\boldsymbol{Y}$, and hence, obtain $\boldsymbol{X}$.

Table 15.4 Codewords of a (7, 4) Hamming code

Massage word	codewords	weight of codeword
0000	0000000	0
0001	1010001	3
0010	1110010	4
0011	0100011	3
0100	0110100	3
0101	1100101	4
0110	1000110	3
0111	0010111	4
1000	1101000	3
1001	0111001	4
1010	0011010	3
1011	1001011	4
1100	1011100	4
1101	0001101	3
1110	0101110	4
1111	1111111	7

Table 15.5 Decoding table for a (7, 4) Hamming code

Syndrome	error pattern
000	0000000
100	1000000
010	0100000
001	0010000
110	0001000
011	0000100
111	0000010
101	0000001

Ex 15.7

From the data in Ex. 15.5, find the input corresponding to a received signal $Y = 001110$

Solution

We first tabulate all cosets corresponding to all inputs and all errors (Table 15.6) called standard array. Then, we create the syndrome look up table for each error pattern, using eqn. (15-26) as in (Table 15.7).

Table 15.6 Error and Cosets

e	Cosets for different error vectors						
000000	110100	011010	101110	101001	011101	110011	000111
000001	110101	011011	101111	101000	011100	110010	000110
000010	110110	011000	101100	101011	011111	110001	000101
000100	110000	011110	101010	101101	011001	110111	000011
001000	111100	010010	100110	100001	010101	111011	001111
010000	100100	001010	111110	111001	001101	100011	010111
10000	010100	111010	001110	001001	111101	010011	100111
010001	100101	001011	111111	111000	001100	100010	010110

Table 15.7 Error and Syndrome

error	syndrome
000000	000
000001	101
000010	011
000100	110
001000	001
010000	010
100000	100
010001	111

We have found for $\boldsymbol{Y} = [0011110]$ that

$$\boldsymbol{S} = [0\,0\,1\,1\,1\,0]\,\boldsymbol{H}^T = [1\,0\,0]$$

From Table (15.7), the corrected vector and expected error $\hat{e}$

$$\begin{aligned}\boldsymbol{X} &= \boldsymbol{Y} + \hat{\boldsymbol{e}} \\ &= 0\,0\,1\,1\,1\,0 + 1\,0\,0\,0\,0\,0 \\ &= 1\,0\,1\,1\,1\,0\end{aligned}$$

which is the transmitted X

15.6 Convolutional Codes:

Convolutional coders – like block coders – take each set of k bits and put out a set of n bits. These extra bits are used by the receiver to detect and correct bit errors. In block coders, the input is k bit and the output is n bits which appear upon inputting the input bits. In convolutional coders, each k bits block at the input is mapped to an n bit output, but this n bits output depends on the correct k bits block at the input and also on the previous K blocks of k bits that came in before. Consider the coder shown (Fig. 15.5).

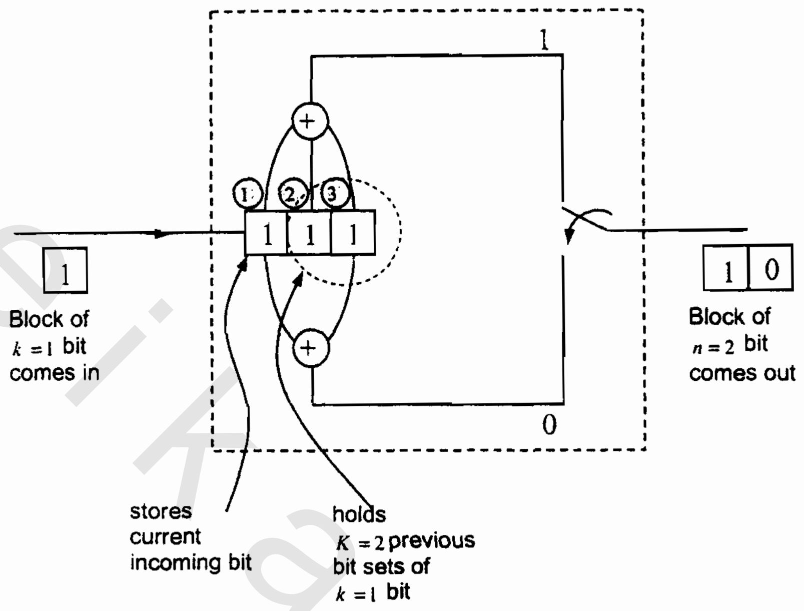

1
1
1
1
1
2
3
1
1
0
Block of $k=1$ bit comes in
Block of $n=2$ bit comes out
0
stores current incoming bit
holds $K=2$ previous bit sets of $k=1$ bit

Fig. (15.5) Convolutional coder

When $k=1$ bit comes in, it is stored in the position "1". Then this bit is moved to position "2", then to position "3". The top adder takes all three bits in the shift register and adds them together (modulo 2), and uses that to create the first bit of the output. The bottom adder takes the first and third bits and adds them together (modulo 2), and this forms the second bit of the output. The output switch acts as parallel to serial converter. Thus, for $k=1$ input the output depends on this input and $K=2$ previous bits. Hence, two output bits are created for every, 1 bit input. Assuming that at $t=0$ we have all O's in the shift register, as clock pulses arrive, the bit is shifted along the shift register. For an input 10100, we have the following output sequence.

Table (15.8) Convolutional coder

Time	Input bit	Shift register contents	Output	
			bit 1	bit 0
0	-	000	-	-
1	1	100	1	1
2	0	010	1	0
3	1	101	0	0
4	0	010	1	0
5	0	001	1	1

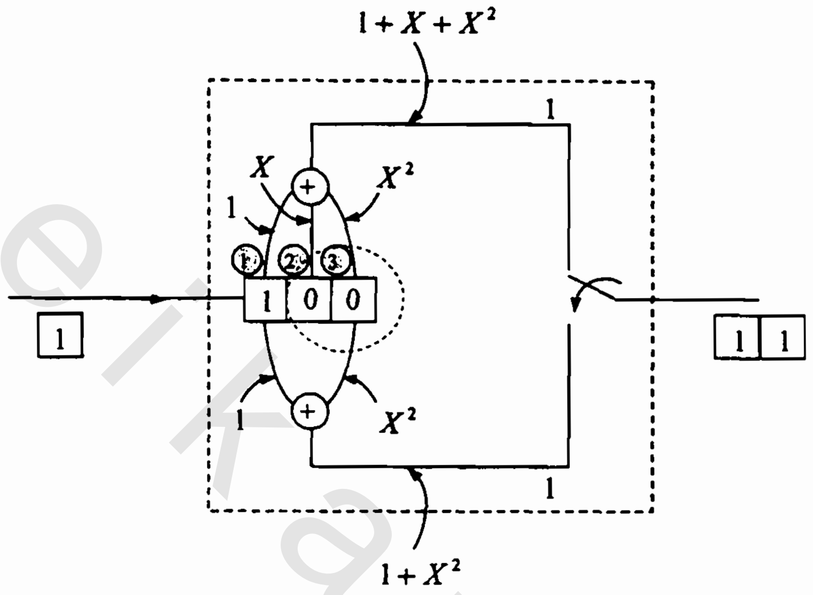


Fig. (15.6) Polynomial representation of convolutional coder

Thus the output is 1110001011 corresponding to the input 10100. We may represent the convolutional coder above by a polynomial

Ex. 15.8

Consider the coder shown (Fig. 15.6), find the polynomials

Solution

$$g_1(X) = 1 + X + X^2$$
$$g_2(X) = 1 + X^2$$

For the input 10100 we multiply the input by the polynomial for each branch

$$m(X) = 1 + 0X + 1X^2 + 0X^3 + 0X^4 = 1 + X^2$$
$$m(X)\ g_1(X) = (1 + X^2)(1 + X + X^2)$$
$$= 1 + X + X^2 + X^2 + X^3 + X^4$$
$$= 1 + X + X^3 + X^4$$

Noting $X^2 + X^2 = 0$

Similarly ,

$$m(X)\ g_2(X) = (1 + X^2)(1 + X^2) = (1 + X^4)$$

15.7 The Trellis Diagram:

This is a way of representing what goes on at the convolutional coder from one time to another. To construct a trellis, we draw a set of times 0,1,.. from left to right. Below each time, we draw a dot for each possible state or node representing the first two positions from the left in the shift register, after a new digit comes in, i.e., what is currently in position 1 and position 2, noting that what is in position 3 will be cast out. Thus, the possible states are 00, 01, 10, 11. There are two types of lines: solid line and dotted line – leaving each state and entering a new state. The dotted line describes what happens when 1 enters at the input. Considering the coder of Fig. (15.5), assume that 1 comes in, and we are at state 00 (0 in position 1 and 0 in position 2). Then, as 1 comes in, the shift register will now contain 100, which means the output will be 11, and the new state (i.e. what is in position 1 and 2) is 10. This is shown in Fig. (15.7a). The line is dotted because 1 is received, whereas if 0 were received the line would be solid. On top of the line is the output after the digit is received. We now add a dotted line and a solid line to each and every dot (state) at each and every time, which leads to the trellis diagram (Fig. 15.7b). It fully describes all possible case in the convolutional coder considered. It tells us what comes out (by looking at the top of at the line), given what was in the shift register position 1 and position 2 (the dot) and the input digit (the connecting line being solid or dotted).

15.8 Channel Decoding

In a channel decoder (Fig. 15.8), a digit sequence $\boldsymbol{m}$ comes in. Considering the coder of Fig. (15.5), with $\boldsymbol{m} = 000$ coming in, this is mapped to the output $\boldsymbol{u} = 00\ 00\ 00$. This output is now sent across the channel through a modulator, the channel and then the demodulator. What comes out from the demodulator and fed to the channel decoder will be called $\mathbf{v}$. If there is no error, then $\boldsymbol{v} = \boldsymbol{u}$. But due to error, $\mathbf{v} = \boldsymbol{u} + e$ where e represents the digit error. Let us assume $e = 00\ 00\ 01$. The goal of the channel decoder is to look into $\mathbf{v}$ and come up with the best guess of $\boldsymbol{m}$, the original information despite the error that has occurred. Let us call the decoder's, guess at $\boldsymbol{m}$ the vector $\boldsymbol{m}'$. The goal is to make $\boldsymbol{m}'$ as close to $\boldsymbol{m}$ as possible. If there is no error, then $00\ 00\ 00$ comes about by following the chain of 00 outputs on top of the solid lines. Now, the received input is $\mathbf{v} = 00\ 00\ 01$. We see that there is no path on the diagram leading to that output. So, the first thing the decoder does is to look at the received vector $\mathbf{v}$ and check if it matches an output path through the trellis. If the answer is no, then there must be an error.

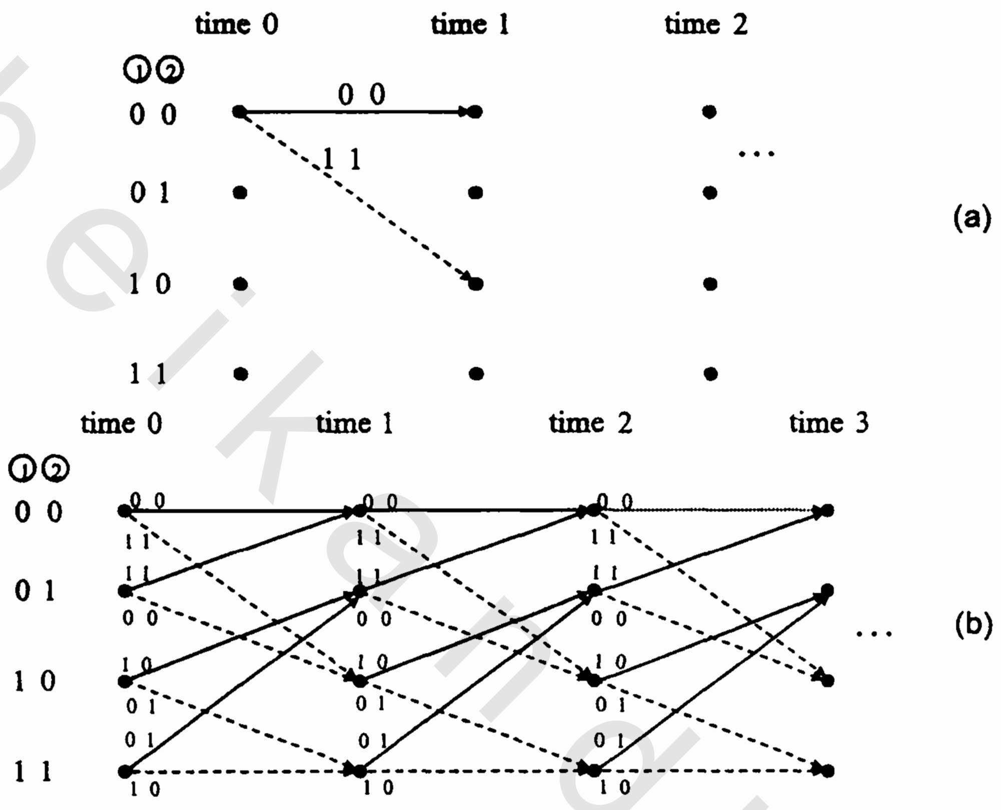


Fig. (15.7) Trellis representation

(a) basic concept *(b) complete trellis for the coder of (Fig. 15.5)*

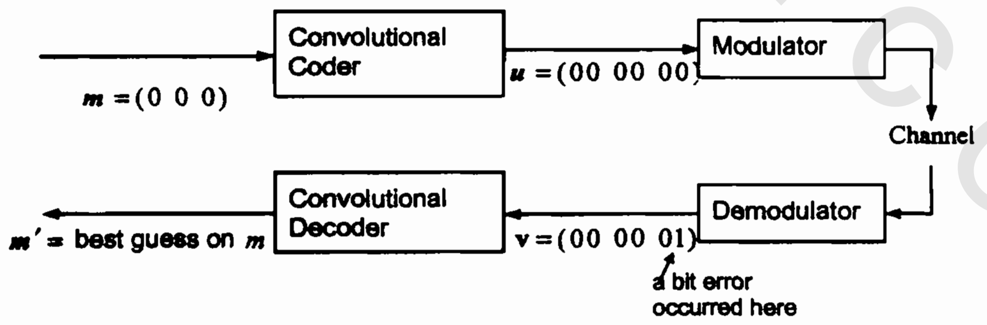


Fig. (15.8) Channel coder-decoder

Thus, the first thing the decoder does is to detect the presence of an error. A convolutional decoder can also correct errors using the trellis diagram by checking what path in the trellis closest to $\mathbf{v}$ in terms of the fewest digit difference. In this case, the closest path to $\mathbf{v} = (00\ 00\ 01)$ is $\boldsymbol{u}' = (00\ 00\ 00)$ as in Fig. (15.9a). Then the channel decoder decides that the sent digits must have been $\boldsymbol{u}' = 00\ 00\ 00$. If these are the sent digits, then the input that created them is $\boldsymbol{m}' = (000)$

There is a simple way to find $\boldsymbol{m}'$ once we have figured out $\boldsymbol{u}'$. All we have to do is look at the trellis at the series of solid and dashed lines **corresponding to** $\boldsymbol{u}'$. A solid line tells us a 0 is the digit in $\boldsymbol{m}'$ and the dashed line is 1 in $\boldsymbol{m}'$.
Using Fig. (15.9b), we see that $\boldsymbol{u}' = 00\ 00\ 00$ is the output path corresponding to the top solid line, of the trellis. On top of the first branch of the path (solid line) there is 00. Thus, the first output digit in $\boldsymbol{m}'$ is 0. For the second 00 branch, it is also solid line, then the second digit in $\boldsymbol{m}'$ is 0 and so on.

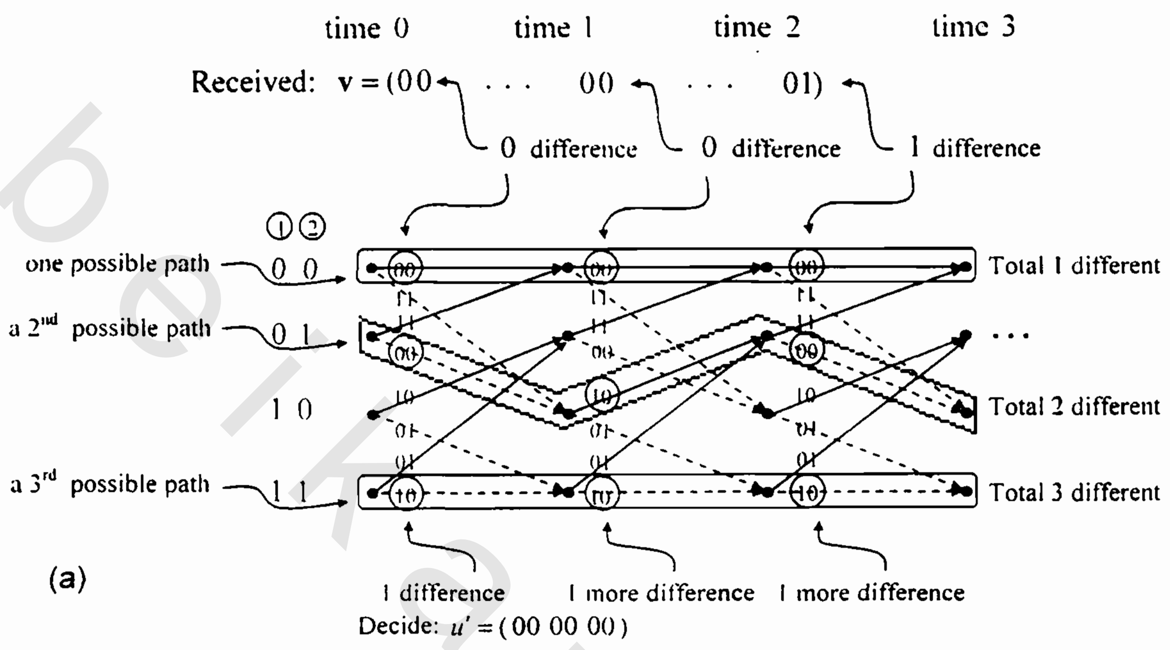
15.9 The Viterbi Algorithm:

Viterbi presented a simple way to look through the trellis. The Viterbi algorithm (VA) lets the computer or DSP chip find the closest path to $\mathbf{v}$ in the trellis easily and effectively. We start at time 0 in the trellis and move from left the right. At each time we systematically eliminate some of the paths. In fact, we eliminate all but 4 of the paths (for number of states=4)

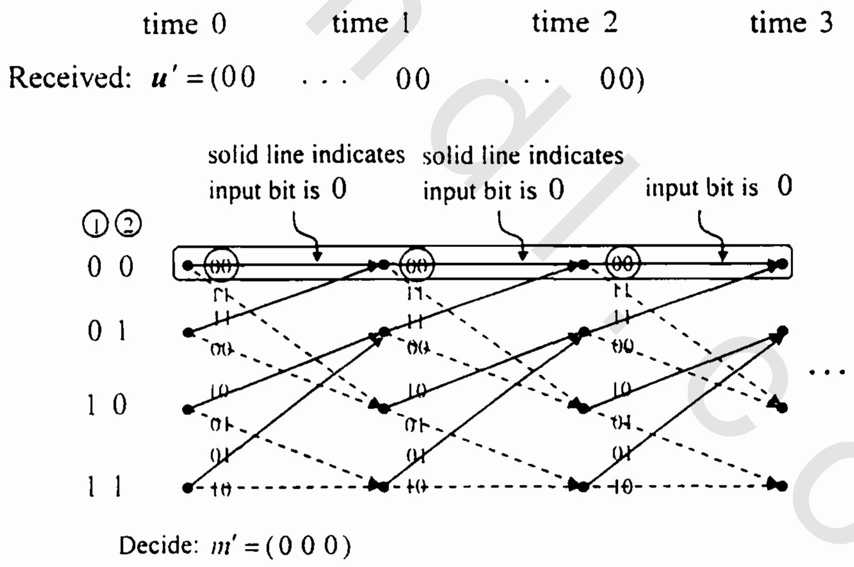
The path elimination may be understood by referring to Fig. (15.7). At time 1 (for initial conditions 0000 at O) we look at the top node. There are 2 paths that head into this node, one originating from parent node A and one from parent node B. VA helps us make a decision on which is the better parent and discard the other. We repeat this at every time.

Consider the coder of Fig. (15.5). Let the input be $\boldsymbol{m} = (00)$ for which the output $u = (00\ 00)$ and the channel decoder receives $\mathbf{v} = (00\ 00)$. To understand how VA works in this example, we associate with each start node the number '0' (Fig. 15.13b). Assume $v = 000001$ (error has occurred). Then we examine for the trellis of Fig.(15.7) the top node at time 1 (Fig. 15.10b). For this node there are 2 possible parent nodes; node 0 at time 0 and node1 at time 0. To decide which is the best parent node, we follow the following procedure:

1. If we started at node 0 (time 0) and moved to node 0 (time 1), the first output digit would be 00. Comparing this to $\mathbf{v}$ when the first two output digits are 00, no error has occurred if parent is node 0 (time 0). We add this 0 to the 0 number that we gave node 0 (time 0) (Fig. 15.10b).



(a)


time 0
time 1
time 2
time 3
Received: u′ = (00 . . . 00 . . . 00)
solid line indicates input bit is 0
solid line indicates input bit is 0
input bit is 0
① ②
0 0
0 1
1 0
1 1
Decide: m′ = (0 0 0)

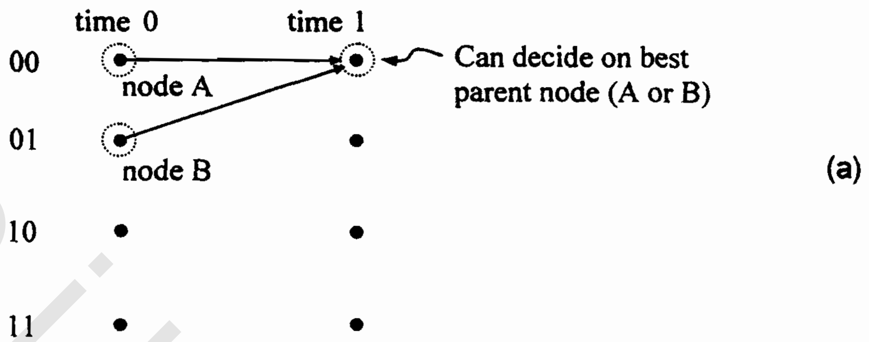
(b)

Fig. (15.9) Using the trellis to get the right input
(a) deciding on closest path *(b) reading out the output digits*

2. If we started at node 1 (time 0) and moved to node 0 (time 1), the first output digit would be 11. Comparing this to **v** when the first two output digits are 00 then there are 2 bit errors if parent is node 1 (time 0). We add this 2 to the 0 number that we gave node 1 (time 0) as in Fig. (15.10b) for a total of 2.

Since starting at node 0 (time 0) and moving to node 0 (time 1) creates the fewest total bit errors (0), we say that the parent node for node 0 (time 1) is node 0 (time 0) and carries with it the number 0 (for zero total errors with this selection as in Fig. (15.10c). We repeat this for node 1 (time 1) (Fig. 15.11a). In Fig. (15.11d), we find that node 2 (time 0) and node 3 (time 0) are possible parent nodes. We decide between them as follows:

1. For node 2 (time 0) as starting node, moving to node 1 (time 1), the output is 10. Comparing this to the first two bits of **v** which are 00, we say 1 bit error if parent node is node2 (time 0). We add this 1 to the 0 number and give the node 2 (time 0) a total of 1
2. For node 3 (time 0) as starting node moving to node 1 (time 1), the output is 01. Comparing this 0 to the first two bits of **v** which are 00, we say that we have 1 bit error if parent node is node 3 (time 0). We add this 1 to the 0 number that we gave node 3 (time 0) for a total of 1.
3. We choose the top node in case of a tie (both totals equal), carrying with it the number 1. We repeat this for node 2 (time1) and node 3 (time 1) (Fig. 15.10b). This covers the first move from left to right through the trellis.
4. We repeat this procedure as we go down the trellis until **v** is exhausted. At the end we choose the node with the least accumulated value according to the path leading to it.



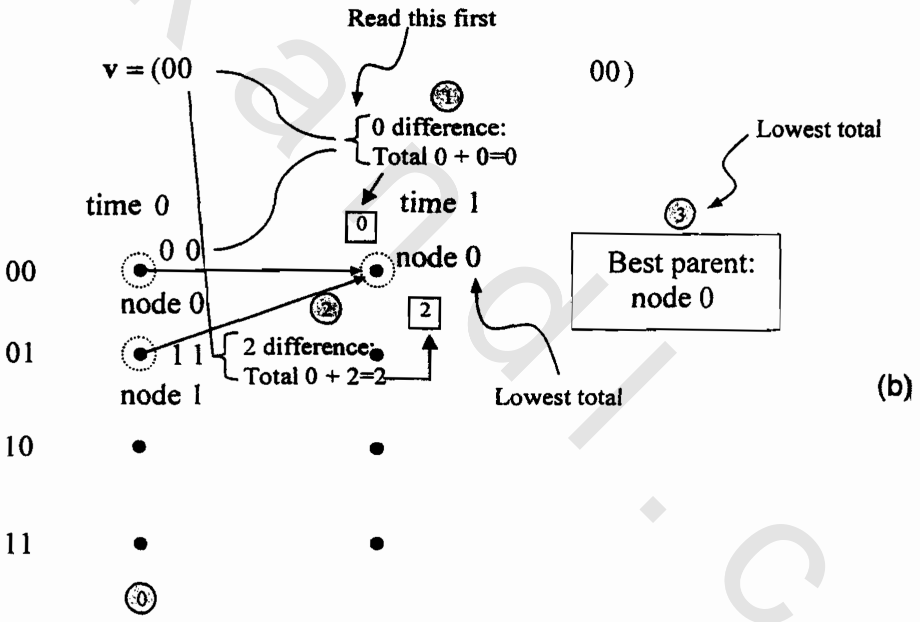
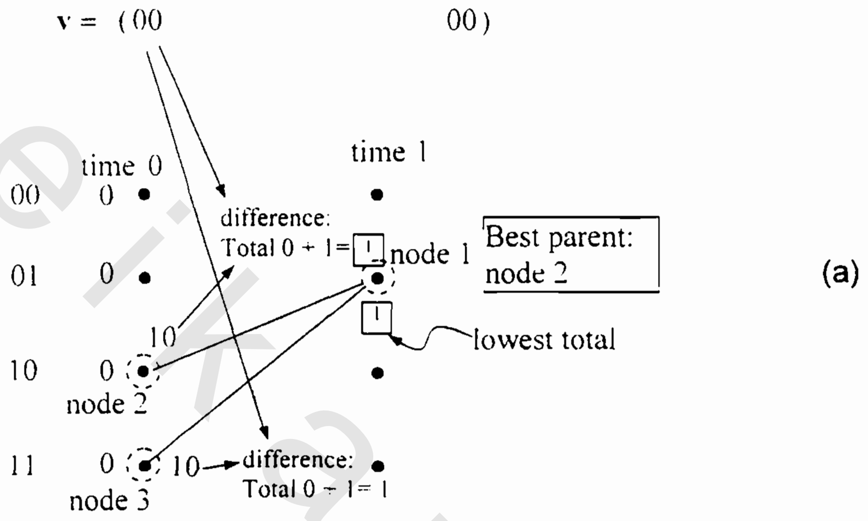


Fig. (15.10) VA steps for node 0 at time 0

a) concept of parent node

b) For initial values 00001 picking the best parent node for node 0 at time 1



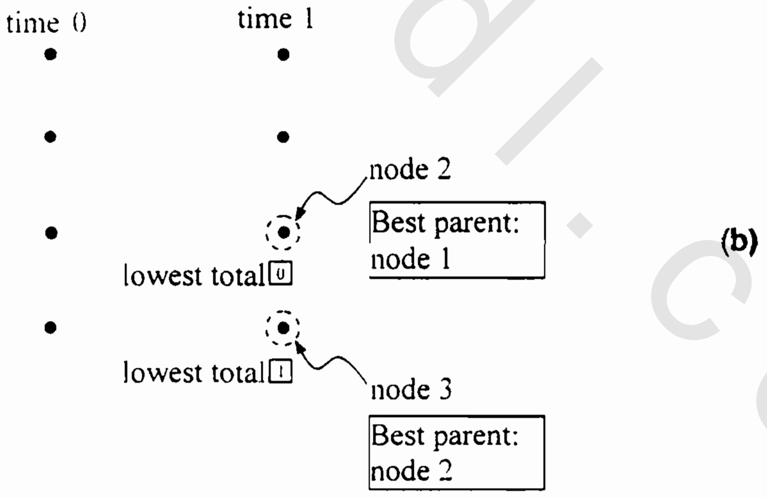


Fig. (15.11) VA steps for nodes 1, 2, 3 at time 1
a) node 1 at time 1 b) node 2 and 3 at time 1

Problems

1. Demonstrate the use of C_3 in Ex. 15.2.

2. Verify eqn. (15-38). Then explain its implication.

3. Fill in the calculations in Ex. 15.6.

4. Implement VA algorithm as outlined in section (15-9) for the trellis of Fig. (15.10), then find the input for $\mathbf{v} = (00\ 00)$

5. Repeat the above problem for $\mathbf{v} = (01\ 00), (10\ 00), (00\ 01)$

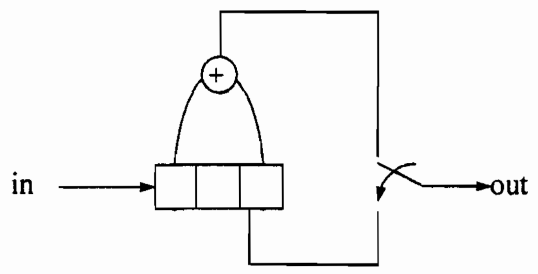
6. Repeat the above problem if $\mathbf{v} = (00\ 01), (00\ 10), (00\ 10)$

7. Determine the polynomial representation for the coder shown.

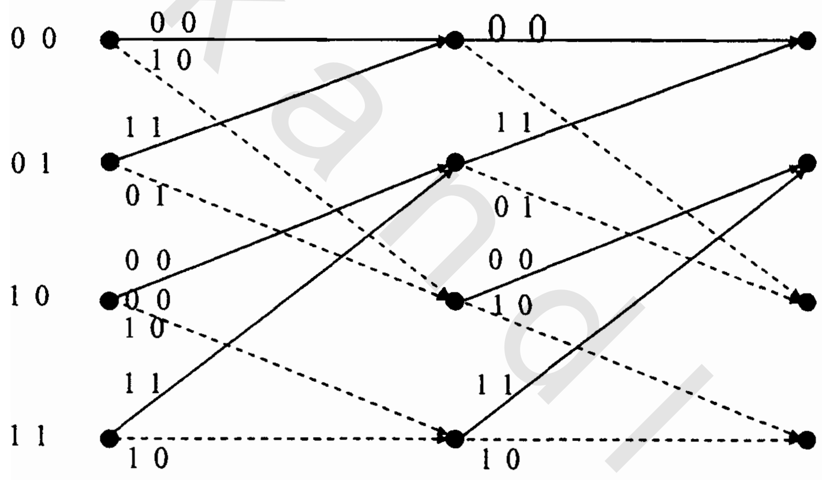
8. Obtain the trellis for the above problem.

9. For the above trellis find the message, for $\mathbf{v} = 0010$.

10.. For the trellis shown find the message if the channel coder receives $\mathbf{v} = 1111$



Prob. 15.7



Prob. 15.10

References

1. "Digital Communications", B. Sklar, 2nd ed., Prentice Hall, Upper Saddle River, N.J., 2001.

2. "Communication System", S. Haykin, 4th ed., J. Wiley, N.J., 2001.

3. "Digital Communications", S. Haykin, J. Wiley, N.J., 1988.

4. "Digital and Analog Communications Systems", L. Couch, 6th ed., Pearson Education, Prentice Hall, Upper Saddle River, N.J., 2001.

5. "Information and Communication for Engineers", M. Usher and C. Guy, McMillan London, 1997.

6. "Telecommunications, Demystified", C. Nasser, LLH Technology Publishing, Eagle Rock, VA, 2001.