UNIT IV

INFORMATION THEORY AND CODING

Measure of information – Entropy – Source coding theorem – Shannon–Fano coding, Huffman Coding, LZ Coding – Channel capacity – Shannon-Hartley law – Shannon's limit – Error control codes – Cyclic codes, Syndrome calculation – Convolution Coding, Sequential and Viterbi decoding

Concept of amount of information(Measure of Information)

1.(a).Write short notes on measure of information.

Or

(b).Describe the expression for amount of information.

The amount of information or messages transmitted over a channel is represented in statistical terms such as probability of occurrence.

Principle:

If the probability of occurrence is more, the amount of information is less. Similarly If the probability of occurrence is less, the amount of information is more.

Assume x_j as an event, so that $p(x_j)$ is the occurance of an event

Then the information due to the event is

$$I(x_j) = log_2\left[\frac{1}{p(x_j)}\right] \quad \text{.....1}$$

Assme x_j and y_k as two independent events

Therefore $I(x_j, y_k) = \log_2\left[\frac{1}{p(x_j, y_k)}\right]$ 2

$$= log_2\left[\frac{1}{p(x_j)}\right] + log_2\left[\frac{1}{p(y_k)}\right] \quad \text{.....3}$$

$$= I(x_j) + I(y_k) \quad \text{..... 4}$$

Where $I(x_j) = log_2\left[\frac{1}{p(x_j)}\right]$ ...5

The above expression indicates that the amount of information is related to the logarithmic of the inverse of the probability of occurance of an event $p(x_j)$

Units of Information

Different units of information can be defined from different bases of algorithms
Base '2'=the unit is bit
Base '10'= the unit is decit

Average Informtaion or Entropy

2.(a).Write the expression for entropy and also its properties.

Or

(b).Give the expression for Average information and als its properties.

Definition

It is defined as the process of producing average information per individual message in a particular interval

Let L be the total message, $m_1, m_2 \ldots . m_k$ represents discrete messages and $p_1, p_2 \ldots . p_k$ represents the probability of discrete messages.

The amount of information in m_1 is

$$I_1 = log_2 \left[\frac{1}{p_1}\right] \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

The total amount of information due to m_1 message is

$$I_{t1} = p_1 L log_2 \left[\frac{1}{p_1}\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

The total information due to L message is

$$I_t = I_{t1} + I_{t2} + \cdots \ldots . I_{tk} \ldots\ldots\ldots\ldots..(3)$$

$$I_t = p_1 L log_2 \left[\frac{1}{p_1}\right] + p_2 L log_2 \left[\frac{1}{p_2}\right] + \cdots \ldots . p_k L log_2 \left[\frac{1}{p_k}\right] \qquad \ldots\ldots\ldots\ldots..(4)$$

The average information per message will be

$$Average\ information = \frac{Total\ information}{Total\ Messages} \qquad \ldots\ldots\ldots\ldots..(5\ )$$

$$Average\ information = \frac{I_t}{L} \quad \text{.....(6)}$$

The average information per message is also called as entropy

$$H(S) = \frac{I_t}{L} \quad \text{.....(7)}$$

$$H(S) = \frac{p_1 L log_2\left[\frac{1}{p_1}\right] + p_2 L log_2\left[\frac{1}{p_2}\right] + \cdots \ldots p_k L log_2\left[\frac{1}{p_k}\right]}{L} \quad \text{.....(8)}$$

$$H(S) = L\left[\frac{p_1 log_2\left[\frac{1}{p_1}\right] + p_2 log_2\left[\frac{1}{p_2}\right] + \cdots \ldots p_k log_2\left[\frac{1}{p_k}\right]}{L}\right] \quad \text{.....(9)}$$

$$H(S) = p_1 log_2\left[\frac{1}{p_1}\right] + p_2 log_2\left[\frac{1}{p_2}\right] + \cdots \ldots p_k log_2\left[\frac{1}{p_k}\right] \quad \text{.....(10)}$$

$$H(S) = \sum_{k=1}^{K} p_k log_2\left[\frac{1}{p_k}\right] \quad \text{.....(11)}$$

The above expression gives the entropy of a discrete memory less source

Properties of entropy.

a) For sure event or impossible event entropy is zero.

b) For M number of equally likely symbols, entropy is $\log_2 M$

c) Upper bound on entropy is H max = $\log_2 M$

source coding theorem or Shannon's first theorem

3.(a).Explain Source Coding theorem

Or

(b).Describe in detail about Shannon's first theorem.

This theorem include Shannon fano coding and Huffman coding to remove redundancy and to improve efficiency

Source coding to improve average information per bit

An important problem in communication system is the efficient representation of data generated by a source.Source encoding or source coding is the process by which the representation is achieved.The device which performs source coding or encoding is called source encoder

Variable length code

It is the encoding process where two codeword's are used.

1. Short codeword is used to represent frequently occurring messages or symbols
2. Longer code word is used to represent rarely occurring symbols

eg) E is more frequently used than Q

Therefore E is represented as a shorter code word and Q is represented as longer code word

Requirement

The efficient source encoder should satisfy the following requirements

1. The code words generated by the encoder should be binary in nature
2. Every code word should represent a single message .

Consider a system as

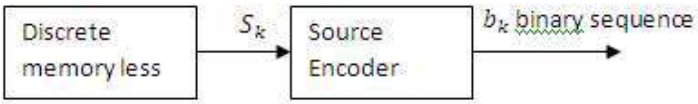


Fig.Transmitter section (Diccrete)

The discrete memory less source produces an output S_k which is converted by the source encoder into a block of 0's and 1's denoted by b_k.The binary codeword for the symbol S_k have length l_k is measured in bits.

The average codeword length

$$\overline{L} = \sum_{k=0}^{K-1} p_k l_k \qquad \text{.....(1)}$$

Where $\bar{L}$ is the average number of bits per source symbol

Let L_{min} denotes the minimum value of $\bar{L}$. Therefore the coding efficiency of the source encoder is

$$\eta = \frac{L_{min}}{\bar{L}} \quad \text{.....(2)}$$

The source encoder is said to be efficient when coding efficiency approaches' unity

Since $\bar{L} \geq L_{min}$ η always less than one. The L_{min} value can be determined by the Shannon's first theorem called source coding theorem.

Statement

Shannon's first theorem is stated as for any distortion less source, a discrete memory less source of entropy H and average codeword length $\bar{L}$ is bounded as

$$\bar{L} \geq H \quad \text{.....(3)}$$

If $L_{min} = H$ then

$$\eta = \frac{H}{\bar{L}} \quad \text{.....(4)}$$

Code Redundancy

It is defined as

$$\gamma = 1 - \eta \quad \text{.....(5)}$$

Code Variance(σ^2)

It is defined as

$$\sigma^2 = \sum_{k=0}^{K-1} p_k \, (n_k - \bar{L}) \quad \text{.....(6)}$$

Where p_k is the probability of k_{th} symbol

n_k is the number of bits assigned to k_{th} symbol

$\bar{L}$ is the average code word length

Source coding techniques

There exist practical methods to design efficient codes. The most used algorithms are:

Shannon-Fano coding,

Huffman coding,

Lempel-Ziv algorithm,

Shannon-Fano codes

4.(a).Explain Shannon fano codes with an example.

Or

(b).with an example,explain Shannon fano coding.

It is based on the following procedure, which can be represented using a tree:

1) Arrange the source messages such as the probabilities are in the descending order
2) Divide the list of messages into two (*Q*) subsets as balanced as possible, in the sense of the sum of elementary
probabilities messages.
3) Assign respectively the symbol "0" and "1", (up to ... *Q-1*) to the first and second(*up to Q-1*) subsets (rootdivided into Q = 2 branches)
4) Repeat the steps 2) 3) with each subset (nodes divided into 2 (or Q) new branches) until that the operation becomes impossible (then each message has become a corresponding code-word left on the tree).

Example: A Discrete memoryless source has 6 symbols s1,s2,s3,s4,s5 and s6 with probabilities 0.4,0.1,0.2,0.1,0.1 and 0.1 respectively. Construct a Shannon Fano Code and also calculate its efficiency.

Solution

Given

Symbols	**Probabilities**
S1	0.4
S2	0.1
S3	0.2
S4	0.1
S5	0.1
S6	0.1

Step 1:

Arrange the given probabilities in decreasing order

Symbols	**Probabilities**
S1	0.4
S3	0.2
S2	0.1
S4	0.1
S5	0.1
S6	0.1

Step 2:

Partitioning

[x1]=[s1,s2]; **[x2]=[s2,s4,s5,s6]**

Or

[x1]=[s1]; [x2]=[s3,s2,s4,s5,s6]

Considering the first partitioning

We have

Symbols	Probability		Stage 1		Stage 2		Stage 3	Code Word	Length
S1	0.4	0.6	0	0.4	0			00	2
S3	0.2		0	0.2	1			01	2
S2	0.1	0.4	1	0.2	0	0.1	0	100	3
S4	0.1		1		0	0.1	1	101	3
S5	0.1		1	0.2	1	0.1	0	110	3
S6	0.1		1		1	0.1	1	111	3

$$\overline{L} = \sum_{k=1}^{K} P_k L_k$$

$$\overline{L} = \sum_{k=1}^{6} P_k L_k$$

$$\overline{L} = P_1L_1 + P_2L_2 + P_3L_3 + P_4L_4 + P_5L_5 + P_6L_6$$
$$\overline{L} = (0.4x2) + (0.2x3) + (0.1x3) + (0.1x3) + (0.1x3) + (0.1x3)$$
$$\overline{L} = 0.8 + 0.4 + 0.3 + 0.3 + 0.3 + 0.3$$
$$\overline{L} = 2.4\ bits/symbol$$

Step 3:
Entropy

$$H(S) = \sum_{j=1}^{J} P(x_j) log_2(\frac{1}{P(x_j)})$$

On substituting the values we get
H(S)=2.32 information bits/symbol

Step 4:
Efficiency

$$\eta = \frac{H}{L}$$

$$\eta = \frac{2.32}{2.4}$$

$$\eta = 96.7\%$$

Problem

1) A discrete memoryless source has symbols x1, x2, x3, x4, x5 with probabilities of 0.4, 0.2, 0.1, 0.2, 0.1 respectively. Construct a Shannon Fano code for the source and calculate code efficiency η .

Solution : Step 1 : Arrange the given probabilities in descending order. given probabilities P1 = 0.4, P2 = 0.2 ,P3 = 0.1 , P4 = 0.2 , P5 = 0.1

Probabilities in descending order

Symbols	Probabilities
x_1	0.4
x_2	0.2
x_3	0.2
x_4	0.1
x_5	0.1

Step 2 : The initial partitioning can be done in two ways (ie) we can split as equiprobable in two methods

Method 1:

Symbol	probability	Stage 1	stage 2	Stage 3	code word	No of bits per message (lk)
x_1	0.4	0	0		00	2
	0.6	0.4				
x_2	0.2	0			01	2
		0.2	1			
x_3	0.2	1			10	2
		0.2	0			
x_4	0.1	1		0	110	3
	0.4	0.2	1			
x_5			0.1	1	111	3
	0.1	1	0.1			

$$\overline{L} = \sum_{k=0}^{K-1} p_k \ell_k = \sum_{k=0}^{4} P_k \ell_k$$

$$= P_0 l_0 + P_1 l_1 + P_2 l_2 + P_3 l_3 + p_4 l_4$$

= (0.4×2) + (0.2×2) + (0.2×2) + (0.1×3) + (0.1×3)

= 0.8 + 0.4 + 0.4 + 0.3 + 0.3

L = 2 bits / symbol

Step 3 : Entropy of the source

$$H(S) = \sum_{k=0}^{K-1} P_k \log_2 \frac{1}{P_K}$$

$$= \sum_{k=0}^{K-1} P_k \log_2 \frac{1}{P_K}$$

$$= P_0 \log_2 \left(\frac{1}{p_o}\right) + P_1 \log_2 \left(\frac{1}{p_1}\right) + P_2 \log_2 \left(\frac{1}{p_2}\right) + P_3 \log_2 \left(\frac{1}{p_3}\right) + P_4 \log_2 \left(\frac{1}{p_4}\right)$$

$$= 0.4 \log_2 \frac{1}{0.4} + 0.2 \log_2 \left(\frac{1}{0.2}\right) + 0.2 \log_2 \left(\frac{1}{0.2}\right) + 0.1 \log_2 \left(\frac{1}{0.2}\right) + 0.1 \log_2 \left(\frac{1}{0.1}\right)$$

$$= 0.4 \frac{log10^{\left(\frac{1}{0.4}\right)}}{log10^2} + 0.2 \frac{log10^{\left(\frac{1}{0.4}\right)}}{log10^2} + 0.2 \frac{log10^{\left(\frac{1}{0.4}\right)}}{log10^2} + 0.1 \frac{log10^{\left(\frac{1}{0.4}\right)}}{log10^2} + 0.1 \frac{log10^{\left(\frac{1}{0.4}\right)}}{log10^2}$$

$$= (0.4 \times 1.3219) + (0.2 \times 2.3219) + (0.2 \times 2.3219) + (0.1 \times 3.3219) + (0.1 \times 3.3219)$$

$$= 0.52876 + 0.46439 + 0.46439 + 0.33219 + 0.33219$$

$$= 2.12192 \text{ bits/symbol.}$$

Efficiency η =(H)/ L = 2.12192/ 2.2 = 0.96450

% η = 96.45 %

Huffman codes

5.(a).Explain Huffman code with an example.

Or

(b).With an example ,write the procedure for Huffman Coding.

It uses the same principle as Shannon Fano coding i.e assigning different number of binary digits to the messages according to their probability of occurrence

This coding method leads to the lowest possible value of $\bar{L}$ for a given message sequence M, resulting in a maximum efficiency.

Hence it is also known as the minimum redundancy code or optimum code**.**

The Huffman code is ***optimum*** in the sense that no other instantaneous code for the same alphabet (and same probabilities distribution) can have a better efficiency (i.e. a lower expected length). In particular, the code efficiency

obtained by Huffman algorithm is greater than or equal to the code efficiency obtained by Shannon-Fano algorithm.

But this efficiency is 1 (optimum code is not necessarily *absolutely optimal* code !).It can be proved that an instantaneous *optimum* code must satisfies the following properties:

- the shorter codewords are assigned to source messages with higher probabilities,24
- lengths of the two (Q) longest (*i.e.* less likely) codewords are equal
- the two (Q) longest codewords differ only in the last symbol (and correspond to the two least likely source messages).

Algorithm is presented in the particular case of a binary code ($Q = 2$) but is easily generalized for Q >2.

It is based on the following procedure, which can be represented by using a tree:

1) Arrange the "outputs" (source messages in the initial iteration) in decreasing order of their probabilities.

2) Combine the two (Q) least probable messages together into a single new "output" (node of the tree) that replaces the two (Q) previous ones, and whose probability is the sum of the corresponding probabilities.

3) If the number of remaining "outputs" is 1 (the remaining node is the root of the tree), then go to the next step;otherwise go to step 1 (and increment the number of iterations), with a new list to be arranged, with a number of "outputs" reduced.

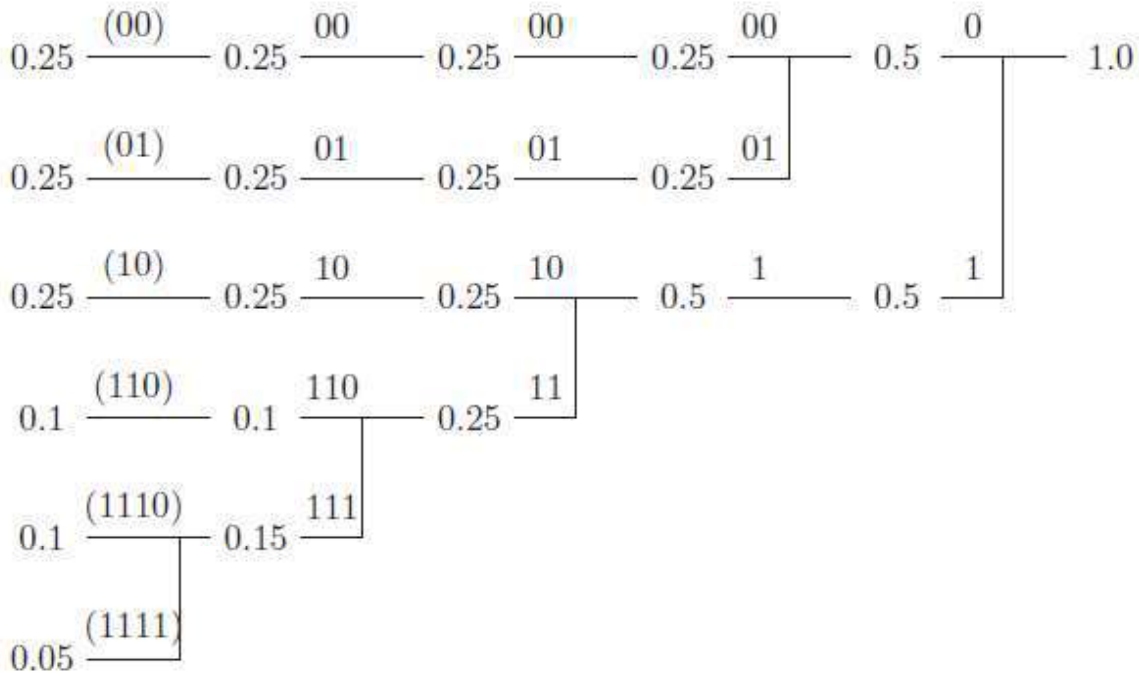
For encoding (assignment of the Q-ary symbols to the different nodes, we have to proceed backward i.e. from the root of the tree (last iteration node) to the different terminal nodes (including first iteration nodes)

4) Assign arbitrary "0" and "1" (up to "Q-1") as first symbol of the 2 (Q) words (nodes) corresponding to the 2 (Q) remaining outputs (last iteration node with sum = 1).

5) If an output is the result of the merger of 2 (Q) outputs in a preceding iteration, append the current word (node) with a "0" and "1" (up to "Q-1") to obtain the word (node) for the preceding outputs and repeat 5). If no output is preceded by another output in a preceding iteration, then stop (first iteration).

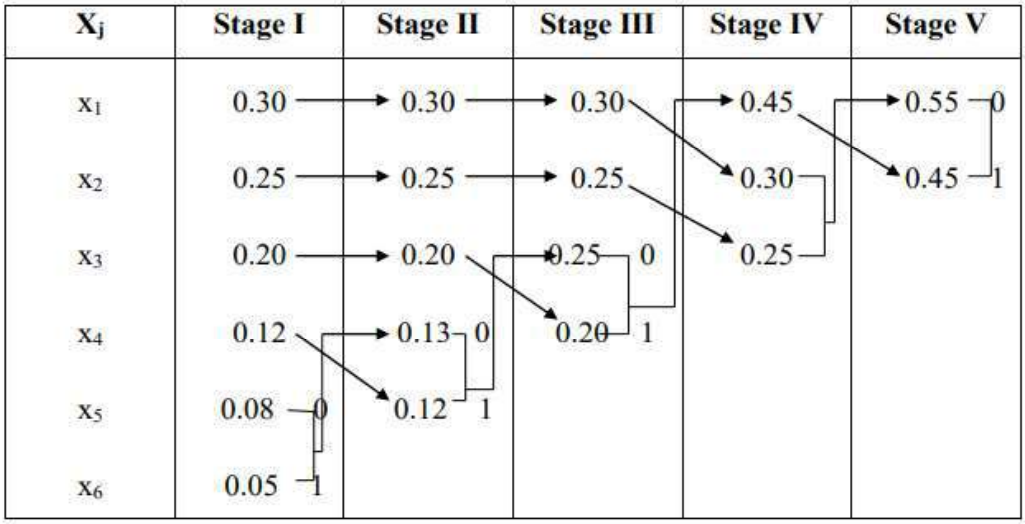
Example Consider a source with alphabet {1, 2, 3, 4, 5, 6} and symbol probabilities 0.25, 0.25, 0.25, 0.1, 0.1 and 0.05, respectively. Obtain the Huffman code

Solution. By following the Huffman encoding procedure, the Huffman code is obtained as 00, 01, 10, 110, 1110, 1111.



Problem

1.A discrete memoryless source has 6 symbols x1, x2, x3, x4, x5, x6 with probabilities 0.30, 0.25, 0.20, 0.12, 0.08, 0.05 respectively construct a Huffman code and calculate its efficiency also calculate redundancy of the code



Message	Probability	Code word	Number of bits k
x_1	0.3	00	2
x_2	0.25	01	2
x_3	0.2	11	2
x_4	0.12	101	3
x_5	0.08	1000	4
x_6	0.05	1001	4

$$\overline{L} = \sum_{k=0}^{K-1} P_k \ell_k \quad = \sum_{k=0}^{5} P_k \ell_k$$

= P0 l0+ P1 l1+P2 l2+P3 l3+P4 l4+ P5 l5

= 0.30× 2 + 0.25×2 +2×2 + 0.12×3 + 0.08×4+0.05×4

= 0.60 + 0.50 + 0.40 + 0.36 + 0.32 +0.020

= 2.38 bits /symbol.

Entropy

$$H(s) = \sum_{k=0}^{K-1} pk \log_2 \frac{1}{pk} = \sum_{k=0}^{5} pk \log_2 \left(\frac{1}{pk}\right)$$

$$= p_0 \log_2\left(\frac{1}{p_o}\right) + p_1 \log_2\left(\frac{1}{p_1}\right) + p_2 \log_2\left(\frac{1}{p_2}\right) + p_3 \log_2\left(\frac{1}{p_3}\right) + p_4 \log_2\left(\frac{1}{p_4}\right) + p_5 \log_2\left(\frac{1}{p_5}\right)$$

$$= 0.30 \log_2\left(\frac{1}{0.30}\right) + 0.25 \log_2\left(\frac{1}{0.25}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right) + 0.12 \log_2\left(\frac{1}{0.12}\right) + 0.08 \log_2\left(\frac{1}{0.08}\right) + 0.05 \log_2\left(\frac{1}{0.05}\right)$$

$$= 0.3 \frac{log10^{\left(\frac{1}{0.30}\right)}}{log10^2} + 0.25 \frac{log10^{\left(\frac{1}{0.25}\right)}}{log10^2} + 0.2 \frac{log10^{\left(\frac{1}{0.2}\right)}}{log10^2} + 0.12 \frac{log10^{\left(\frac{1}{0.12}\right)}}{log10^2} + 0.08 \frac{log10^{\left(\frac{1}{0.08}\right)}}{log10^2} + 0.05 \frac{log10^{\left(\frac{1}{0.05}\right)}}{log10^2}$$

$$= 0.0521 + 0.5 + 0.4643 + 0.367 + 0.2915 + 0.216$$

$$= 2.3598 \text{ bits of information /message}$$

Code efficiencyη = (H)/ *L*

= 2.3598/ 2.38

= 0.99

% η = 99 %

Redundancy of code (γ) γ = 1- η

=1-0.99

= 0.01

Problem

A discrete memoryless source (DMS) has fire symbols x1, x2, x3 and x4 with p(x0) = 0.4 ,p(x1) = 0.19, p(x2) = 0.16 ,p(x3) = 0.15, p(x4) = 0.1 (i) Construct the Shannon fano code for x and calculate efficiency of the code (ii) repeat for the Huffman code and compare the results

Shannon Fano code

Symbol	probability	Stage 1	Stage 2	Stage 3	Code word	No of bits per message (ℓ_k)
x0	0.4	0	0	0	00	2
x1	0.19	0	1	1	01	2
----------	----------	----------	---------		10	2
x2	0.16	1	0		110	3
x3	0.15	1	1		111	3
x4	0.1	1	1			

Entropy

$$H(s) = \sum_{k=0}^{K-1} pk \log_2 \frac{1}{pk} = \sum_{k=0}^{4} pk \log_2 \left(\frac{1}{pk}\right)$$

$$= p_0 \log_2\left(\frac{1}{p_o}\right) + p_1 \log_2\left(\frac{1}{p_1}\right) + p_2 \log_2\left(\frac{1}{p_2}\right) + p_3 \log_2\left(\frac{1}{p_3}\right) + p_4 \log_2\left(\frac{1}{p_4}\right)$$

$$= 0.4 \log_2\left(\frac{1}{0.4}\right) + 0.19 \log_2\left(\frac{1}{0.19}\right) + 0.16 \log_2\left(\frac{1}{0.16}\right) + 0.15 \log_2\left(\frac{1}{0.15}\right)$$

$$+ 0.1 \log_2\left(\frac{1}{0.1}\right)$$

H(s) = 2. 15 bits / message

Codeword length

$$\bar{L} = \sum_{k=0}^{K-1} pk\ell k = \sum_{k=0}^{5} pk\ell k$$

$$= P_0 l_0 + P_1 l_1 + P_2 l_2 + P_3 l_3 + P_4 l_4$$

$$= 0.4 \times 2 + 0.19 \times 2 + 0.16 \times 2 + 0.15 \times 3 + 0.1 \times 3$$

L = 2.25 bits /symbol

Code efficiency = $H(s)/L = 2.15/2.25 = 0.956$

%η = 95.6 %

Huffman code :

Huff man code is constructed as follows

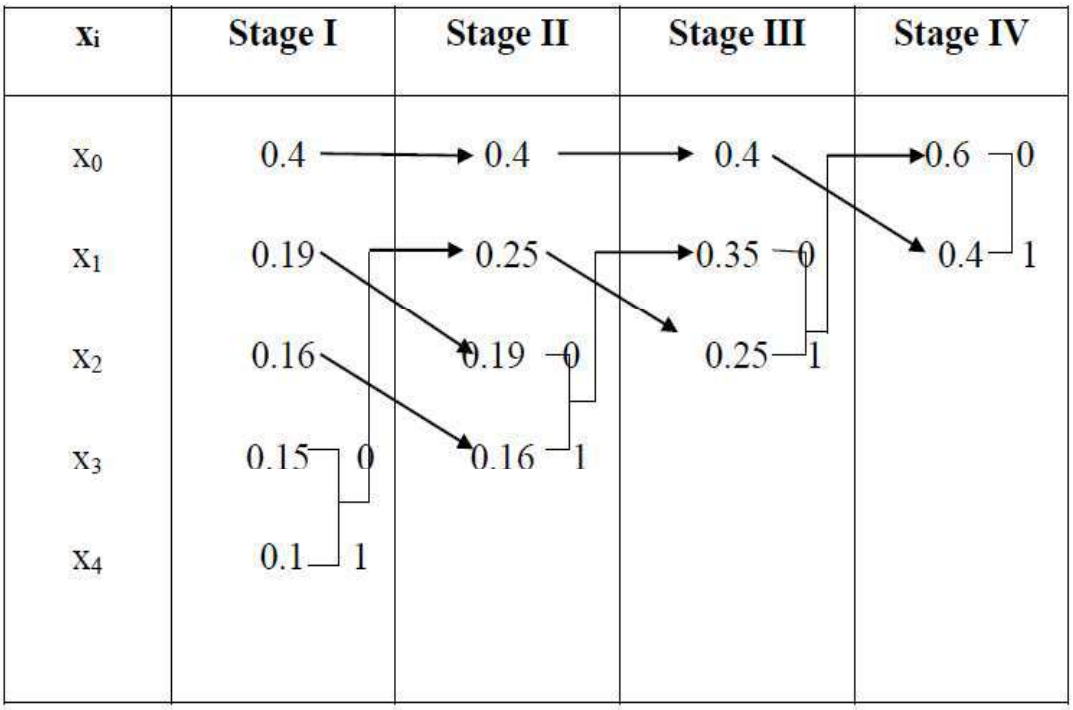
Average code word Length ($\bar{L}$)

$$\bar{L} = \sum_{k=0}^{K-1} p_k \ell_k$$

$$= P_0 l_0 + P_1 l_1 + P_2 l_2 + P_3 l_3 + P_4 l_4$$

$$= 0.4 \times 1 + 0.19 \times 3 + 0.16 \times 3 + 0.15 \times 3 + 0.1 \times 3$$

$$\bar{L} = 2.2 \text{ bits /symbol}$$



Symbol	P(xi)	Codeword	Length (ℓk)
x0	0.4	1	1
x1	0.19	000	3
x2	0.16	001	3
x3	0.15	010	3
X4	0.1	011	3

Entropy

$$H(s) = \sum_{k=0}^{K-1} pk \log_2 \frac{1}{pk} = \sum_{k=0}^{4} p_k \log_2 \frac{1}{p_k}$$

$$= p_0 \log_2 \frac{1}{p_0} + p_1 \log_2 \frac{1}{p_1} + p_2 \log_2 \frac{1}{p_2} + p_3 \log_2 \frac{1}{p_3} + p_4 \log_2 \frac{1}{p_4}$$

$$= 0.4 \log_2 \frac{1}{0.4} + 0.19 \log_2 \frac{1}{0.19} + 0.16 \log_2 \frac{1}{0.16} + 0.15 \log_2 \frac{1}{0.15} + 0.1 \log_2 \frac{1}{0.1}$$

Code efficiency (ɳ)

$$ɳ = \frac{H(s)}{L} = \frac{2.15}{2.2} = 0.977$$

$$\% ɳ = 97.7\ \%$$

Lempelziv Coding

Drawback of Huffman code is that

(i) It requires knowledge of the probabilistic model of the source

(ii) With modeling text, the storage requirements prevent the Huffman coding from capturing the higher-order relationship between words

Principle:

Encoding is done by " Parsing the source data stream into segments that are shortest subsequences not encountered previously.

Example:

6.(a).*Find the encoded blocks for an input binary sequence 1111100000 using Lempel Ziv Code*

Or

(b).Using Lempel Ziv Code,find the encoded blocks for an input binary sequence 1111100000 using Lempelziv code.

Encoder:

Step 1:

Assume that the binary symbols 0 and 1 are already stored in that order in the code book. It can be written as,

Subsequences stored: 0, 1

Data to be parsed:11111000000

Step 2:

Assume the encoding process begins at the left, the shortest subsequences from the left other than 0 & 1 is 11. It is stored as

Subsequences stored: 0, 1,11

Data to be parsed:11100000

Step 3:

The shortest subsequences other than 0, 1,11 is 111. It is shown as

Subsequences stored: 0, 1,11,111

Data to be parsed:00000

Step 4:

The shortest subsequences which is not encountered previously is

Subsequences stored: 0, 1,11,111,00

Data to be parsed:000

Step 5:

The shortest subsequences found according to Lempel Ziv Principle is

Subsequences stored: 0, 1,11,111,00,000

Data to be parsed:…..

Since there is no data to be parsed, the code of binary sequences is shown as

Code Book: 0,1,11,111,00,000

The encoded block has been generated by the following row arrangement.

Numerical positions	1	2	3	4	5	6
Code Book Subsequences	0	1	1 1	11 1	0 0	00 0
Numerical Representation			2 2	3 2	1 1	5 1
Binary Encoded Blocks			010 1	011 1	001 0	101 0

Problems

1. **Calculate amount of information if** $P_k = 1/2$

Solution:

Given:

$$P_k = 1/2$$

Amount of Information $\boldsymbol{I_k = log_2(\frac{1}{P_k})}$

$$I_k = log_2(\frac{1}{1/2})$$

$$I_k = log_2 2$$

$$\boldsymbol{I_k = 1\ bit}$$

2.Calculate the total amount of information for binary symbols '0' and '1' which is transmitted with probabilities ¾ and ¼ respectively, also compare its amount of information.

Solution:

$$P_1 = 3/4$$

$$P_2 = 1/4$$

$$\boldsymbol{I_1 = log_2(\frac{1}{P_1})}$$

$$I_1 = log_2(\frac{1}{3/4})$$

$$I_1 = log_2(\frac{4}{3})$$

$$I_1 = \frac{log_{10}(\frac{4}{3})}{log_{10}(2)}$$

$$\boldsymbol{I_1 = 0.4150\ bit}$$

$$I_2 = log_2(\frac{1}{P_2})$$

$$I_2 = log_2(\frac{1}{1/4})$$

$$I_2 = log_2(4)$$

$$I_2 = \frac{log_{10}4}{log_{10}2}$$

$$\boldsymbol{I_2 = 2\ bits}$$

3..A source s emits symbolsms1,s2,s3 with probabilities 0.25,0.5 and 0.25. Find entropy of a source S

Solution:

$$\boldsymbol{H(S) = \sum_{k=1}^{K} P(x_j) log_2(\frac{1}{P(x_j)})}$$

$$H(S) = P_1 log_2\left(\frac{1}{P_1}\right) + P_2 log_2\left(\frac{1}{P_2}\right) + P_3 log_2\left(\frac{1}{P_3}\right)$$

$$H(S) = 0.25 log_2\left(\frac{1}{0.25}\right) + 0.5 log_2\left(\frac{1}{0.5}\right) + 0.25 log_2\left(\frac{1}{0.25}\right)$$

$$H(S) = 0.25\frac{log_{10}\left(\frac{1}{0.25}\right)}{log_{10}2} + 0.5\frac{log_{10}\left(\frac{1}{0.5}\right)}{log_{10}2} + 0.25\frac{log_{10}\left(\frac{1}{0.25}\right)}{log_{10}2}$$

$$H(S) = 0.25(2) + (0.5)(1) + (0.25)(2)$$

$$\boldsymbol{H(S) = 1.5\ bits/symbol}$$

4.An analog signal is band limited to 250 Hz and sampled at Nyquist rate. The samples are quantized into 4 levels. The probabilities of occurrence of these levels are p_1=p_3=1/8 and p_2=p_4=3/8. Find out the information rate of a source.

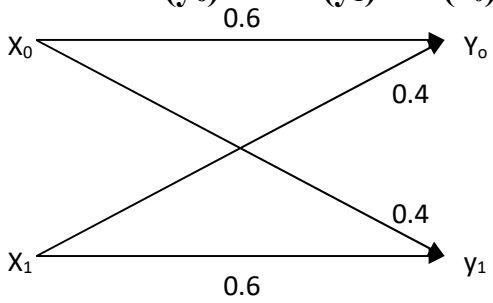
Solution:

Information rate $\boldsymbol{R = rH}$

$$r = 2B$$
$$r = 2x250$$
$$r = 500\ levels/sec$$
$$\boldsymbol{H(S) = \sum_{k=1}^{K} P(x_j) log_2 (\frac{1}{P(x_j)})}$$
$$H(S) = P_1 log_2 \left(\frac{1}{P_1}\right) + P_2 log_2 \left(\frac{1}{P_2}\right) + P_3 log_2 \left(\frac{1}{P_3}\right) + P_4 log_2 \left(\frac{1}{P_4}\right)$$
$$H(S) = \frac{1}{8} log_2(8) + \left(\frac{3}{8}\right) log_2 \left(\frac{8}{3}\right) + \left(\frac{1}{8}\right) log_2(8) + (\frac{3}{8}) log_2 \left(\frac{8}{3}\right)$$

$$H(S) = 0.375 + 0.5306 + 0.375 + 0.5306$$
$$\boldsymbol{H(S) = 1.8113\ bits/symbol}$$

5.A binary symmetric channel is shown in fig. Find the channel matrix of the resultant channel.Find P(y_o) and P(y_1) if P(x_0)=0.3,P(x_1)=0.7



Solution:

Channel Matrix

$$\boldsymbol{P\left(\frac{Y}{X}\right) = \begin{bmatrix} P(\frac{y_o}{x_o}) & P(\frac{y_1}{x_o}) \\ P(\frac{y_o}{x_1}) & P(\frac{y_1}{x_1}) \end{bmatrix}}$$

Channel Matrix $\boldsymbol{P\left(\frac{Y}{X}\right) = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix}}$

$$\boldsymbol{P(Y) = P\left(\frac{Y}{X}\right) P(X)}$$

$$P(Y) = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix} [0.3 \quad 0.7]$$

$$P(Y) = [0.18 + 0.28 \qquad 0.120.42]$$

$$P(Y) = [0.46 \quad 0.54]$$

We know that $P(Y) = [P(y_o)\ P(y_1)]$

$P(y_o) = 0.46 \quad P(y_1) = 0.54$

Channel Capacity

7.(a).Write short notes on Channel Capacity

Or

(b).Describe Channel capacity with necessary equations.

It is defined as the maximum mutual information $I(X;Y)$ in any single use of the channel, where the maximization is over all possible input probability distributions $\{p(x_j)\}$ on X.

Where C is measured in bits per channel use or bits per transmission

$$C = \max_{\{p(x_j)\}} I(X;Y) \qquad \text{.....(1)}$$

The calculation of C involves maximization of mutual information $I(X;Y)$ over j variables $p(x_0), p(x_0), p(x_0), \ldots . p(x_0)$. It must satisfy two conditions

(i) $p(x_j) \geq 0\ for\ all\ j$

(ii) $\sum_{j=1}^{J} p(x_j) = 1$

Channel efficiency

The transmission efficiency or channel efficiency is defined as

$$\eta = \frac{Actual\ transmission}{Maximum\ transmission} \qquad \text{.....(2)}$$

$$\eta = \frac{I(X;Y)}{MaxI(X;Y)} \qquad \text{.....(3)}$$

$$\eta = \frac{I(X;Y)}{C} \qquad \text{.....(4)}$$

Channel Redundancy

The redundancy of the channel is defined as

$$R = 1 - \eta \qquad \text{.....(5)}$$

$$R = 1 - \frac{I(X;Y)}{C} \qquad \text{.....(6)}$$

Noise free channel

We know that mutual information

$$I(X;Y) = H(X) - H(\tfrac{X}{Y}) \qquad \text{.....(7)}$$

The mutual information for a noise free channel is given as

$I(X;Y) = H(X)$ .....(8)

The channel capacity is $C = maxI(X;Y)$

$C = maxH(X)$ .....(9)

$C = log_2 k\ bits/message$ .....(10)

Symmetric Channel

A symmetric channel is defined as one, which has two properties

(i) $H\left(\frac{Y}{x_j}\right) is\ independent\ of\ k$

(ii) $\sum_{j=1}^{J} p(\frac{y_k}{x_j})$ is independent of j

The mutual information for a symmetric channel is

$I(X;Y) = H(Y) - H(\frac{Y}{X})$ .....(11)

$I(X;Y) = H(Y) - \sum_{j=1}^{J} H(Y/x_j)p(x_j)$ .....(12)

$I(X;Y) = H(Y) - A\sum_{j=1}^{J} p(x_j)$ .....(13)

Where

$A = H(\frac{Y}{x_j})$ .....(14)

We know that

$\sum_{j=1}^{J} p(x_j) = 1$ .....(15)0

Therefore

I $(X;Y) = H(Y) - A$ .....(16)

The channel capacity is

$C = max\ I(X;Y)$ .....(17)

$C = max\ (H(Y) - A)$ .....(18)

$C = max\ (H(Y)) - A$ .....(19)

$C = log_2 K - A)$bits/message .....(20)

Binary Symmetric Channel(BSC)

The channael capacity for binary symmetric chnanle is

$C = log_2 K - A)$bits/message .....(21)

$C = log_2 K - H(\frac{Y}{x_j}))$ .....(22)

$C = log_2 K - H(\frac{Y}{x_j}))$ .....(23)

$C = log_2 K - \sum_{k-1}^{2} p(y_k/\ x_j)\ log_2 \frac{1}{p(y_k/\ x_j)}$ .....(24)

Shannon's third theorem or Information capacity theorem or Shannon Hartley theorem

8.(a). Explain Shannon Hartley theorem.

Or

(b).Derive the expression for Channel Capacity theorem.

Here idea of mutual information is used to find the information capacity of band limited Gaussian channels. Consider a zero mean stationary process $X_k(t)$ which is band limited to B hertz.

Let X_k be a constant random variable which is uniformly sampled at the nyquist rate of 2B samples/sec. These samples are transmitted in T sec over a noisy channel. Therefore the number of samples K=2BT.

The channel output is disturbed by additive white Gaussian noise of zero mean and power spectral density $N_o/2$.

Output $Y_k=X_k+N_k$ 5.74

The power transmitted from the input is

$E[X_k^2]=P$ 5.75

Definition

It is defined as the maximum of mutual information between the channel input X_k and channel output Y_k over all possible distributions on the input X_k that satisfy the power constraint $E[X_k^2]=P$

Let $I(X_k;Y_k)$ be the mutual information. Mutual information can be expressed as

$I(X_k;Y_k)=H(X_k)-H(X_k/Y_k)$ 5.76

We have

$H(N_k)= H(X_k/Y_k)= H(Y_k/X_k)$ 5.77

Where $H(Y_k/X_k)$ is the conditional entropy

$H(N_k)$ is the differential entropy of N_k

Therefor e

$I(X_k;Y_k)=H(Y_k)- H(N_k)$ 5.78

The information capacity can be evaluated using the following points

1. The variance of sample Y_k of the received signals is $P+\sigma^2$. Hence differential entropy of Y_k is
   $H(Y_k)=1/2 \log_2[2\pi e(P+\sigma^2)]$ 5.79
2. The variance of sample $N_k= \sigma^2$. Hence differential entropy of N_k is
3. $H(N_k)=1/2 \log_2 [2\pi e(\sigma^2)]$

Therefore

$I(X_k;Y_k)=H(Y_k)- H(N_k)$ 5.80

$I(X_k;Y_k)= 1/2 \log_2[2\pi e(P+\sigma^2)]- 1/2 \log_2 [2\pi e(\sigma^2)]$ 5.81

$$I(Xk; Yk) = \log_2[\frac{2\pi e(P+\sigma^2)}{2\pi e(\sigma^2)}] \quad 5.82$$

$$I(Xk; Yk) = \frac{1}{2}\log_2[1 + \frac{P}{\sigma^2}] \text{ bits/ transmission} \quad 5.83$$

Information capacity can be expressed as

$$C = Blog_2\left[1 + \frac{P}{N_oB}\right]\text{bits/sec} \quad 5.84$$

$$C = Blog_2\left[1 + \frac{S}{N}\right]\text{bits/sec} \quad 5.85$$

It ha s three important parameters

(i) Channel bandwidth
(ii) Average transmitted power
(iii) Noise power spectral density at the output

BANDWIDTH S/N TRADE OFF

Channel capacity of a gaussian channel is given as

$$C = Blog_2[1+\frac{S}{N}]bits/sec. \quad 5.86$$

Above equation indicates that channel capacity depends on two factors

1.Bandwidth of the channel

2.Signal to noise ratio

7 Error Control Coding

Error Control Codes:

9.(a)Write short notes on Error Control Codes.

Or

(b).What is the need for error control codes.

Error control coding aims at developing methods for coding tocheck the correctness of the bit stream transmitted.The bit stream representation of a symbol is called the codeword of that symbol.

Different error control mechanisms:

- Linear Block Codes
- Repetition Codes
- Convolution Codes
- Cyclic codes

Some important terms

- Codeword: The encoded block of n bits is called a codeword.It consists of message bits and redundant bits
- Block length: The number of bits 'n' after coding is called as block length of the code
- Code rate: The ratio of message bits(k) and encoded output bits(n) called code rate(r)

  r = k/n
- Channel data rate: It is the bit rate at the output of the encoder is R_s then channel data rate(i.e.,) the output of encoder be,

  Channel data rate (R_0) = $\frac{n}{k} R_s$
- Code vectors: In block coding,binary information sequencesequence is segmented into message blocks of fixed length,each message block consists of k information bits

  There are a total of 2^k distinct messages.The encoder according certain rule,transforms each input message into binary n-tuple with n>k.The binary n-tuplereferred to as the codeword (or) code vector of the message.It is simpler to visualize 3 bit code words[example 000 to 111]
- Hamming distance: Hamming distance between two code vectors is defined as the number of position in which they differ.

  For example, X = 110 and Y = 101.The two code vectors differ in second and third bits.Hence Hamming distance between X and Y is 2 (i.e) d(x,y) = d = 2

- Minimum Distance (dmin):

  The minimum distance of a linear block code is defined as the smallest hamming distance between any pair of codewords in the code (or) minimum distance is the same as the smallest hamming weight of the difference between any pair of codewords.

  The following table list some of the requirements of error capability of the code

1	Detect upto ‘s’ errors per word	dmin ≥ s+1
2	Correct upto ‘t’ errors per word	dmin≥2t+1
3	Correct upto’ t ‘errors and detect s>t errors per word	dmin≥(t+s+1)

10.(a)Obtain the code for an (n,k) linear cyclic code and explain its working.

Or

(b).Explain in detail about the systematic and non systematic generator matrix in cyclic code

Cyclic codes are the sub class of linear block codes. Cyclic codes can be in systematic or nonsystematic form. In systematic form, check bits are calculated separately and the code vector is X= (M:C) form. Here 'M' represents message bits and 'C' represents check bits.

Definition of Cyclic Code

A linear code is called cyclic code if every cyclic shift of the code vector produces some other code vector. This definition includes two fundamental properties of cyclic codes.

Properties of Cyclic Codes

As defined above, cyclic codes exhibit two fundamental properties :

1. Linearity and 2. Cyclic property

Linearity Property

This property states that sum of any two codewords is also a valid codeword. For example let X_1 and X_2 are two codewords. Then,

$$X_3 = X_1 \oplus X_2$$

Here X_3 is also a valid codeword. This property shows that cyclic code is also a linear cod **Cyclic Property**

Very cyclic shift of the valid code vector produces another valid code vector. Because of this property, the name 'cyclic' is given. Consider an n-bit code vector as shown below:

$$X = \{x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0\}$$

Here $x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0$ etc. represent individual bits of the code vector 'X'. Let us shift the above code vector cyclically to left side. i.e.,

$$one\ cyclic\ shift\ of\ X\ gives, \qquad X' = (x_{n-2}, x_{n-3}, \ldots\ldots, x_1, x_0, x_{n-1})$$

Here observe that every bit is shifted to left by one position. Previously x_{n-1} was MSB but after left cyclic shift it is at LSB position. Here the new code vector is X' and it is valid code vector. One more cyclic shift yields another code vector X". i.e.,

$$X'' = (x_{n-3}, x_{n-4}, \ldots\ldots x_1, x_0, x_{n-1}, x_{n-2})$$

Here observe that x_{n-3} is now at MSB position and x_{n-2} is at LSB position.

Algebraic Structures of Cyclic Codes

The codewords can be represented by a polynomial.

For example, consider the n-bit codeword,

$$X = (x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0\ )$$

This codeword can be represented by a polynomial of degree less than or equal to (n-1). i.e.,

$$X(p) = x_{n-1}p^{n-1}, x_{n-2}p^{n-2}, \ldots\ldots, x_1 p, x_0$$

Here $X(p)$ is the polynomial of degree (n-1)

P is the arbitrary variable of the polynomial

The power of 'p' represents the positions of the codeword bits. i.e.,

p^{n-1} represents MSB

p^0 represents LSB

p^1 represents second bit from LSB side.

Why to represent codewords by a polynomial?

Polynomial representation is used due to following reasons:

i) These are algebraic codes. Hence algebraic operations such as addition, multiplication, division, subtraction etc. becomes very simple.

ii) Positions of the bits are represented with the help of powers of p in a polynomial.

Generation of Code vectors in Nonsystematic Form

Let $M = \{m_{k-1}, m_{k-2}, \ldots m_1, m_0\}$ be 'k' bits of message vector. Then it can be represented by the polynomial as,

$$M(p) = m_{k-1}\, p^{k-1} + m_{k-2}\, p^{k-2} + \cdots + m_1 p = m_0$$

Let $X\ (p)$ represent the codeword polynomial. It is given as,

$$\boxed{X(p) = M(p)G(p)}$$

Here $G(p)$ is the generating polynomial of degree 'q'. For an *(n,k)* cyclic code, *q* = n - *k* represent the number of parity bits. The generating polynomial is given as,

$$G(p) = p^q + g_{q-1}p^{q-1} + \cdots + g_1 p + 1$$

Here $g_{q-1}, g_{q-2}, \ldots g_1$ are the parity bits.

If $M_1\ M_2.\ M_3$ etc are the other message vectors, then the corresponding code vectors can be calculated as

$$X_1(p) = M_1(p)G(p)$$

$$X_2(p) = M_2(p)G(p)$$

$$X_3(p) = M_3(p)G(p)\ \ and\ so\ on$$

All the above code vectors $X_1, X_2, X_3 \ldots$ are in nonsystematic form and they satisfy cyclic property. Note the generator polynomial $G(p)$ remains the same for all code vectors.

Generation of Code vectors in Systematic Form

Now let us study systematic cyclic codes. The systematic form of the block code is,

$$X = (k\ message\ bits : (n-k) check\ bits)$$

$$= (m_{k-1}m_{k-2} \ldots m_1 m_0 : c_{q-1}c_{q-2} \ldots c_1 c_0)$$

Here the check bits form a polynomial as,

$$C(p) = c_{q-1}p^{q-1}c_{q-2}p^{q-2} \ldots c_1 p c_0$$

The check bit polynomial is obtained by

$$\boxed{C(p) = rem\left[\frac{p^q M(p)}{G(p)}\right]}$$

Above equation means -

i) Multiply message polynomial by p^d.

ii) Divide $p^q M(p)$ by generator polynomial.

iii) Remainder of the division is $C(p)$.

Generator and Parity Check Matrices of Cyclic Codes

Non systematic Form of Generator Matrix

Since cyclic codes are subclass of linear block codes, generator and parity check matrices can also be defined for cyclic codes. The generator matrix has the size of k x n. That means there are $'k'$ rows and 'n' columns. Let the generator matrix $G(p)$ be given by,

$$G(p) = p^q + g_{q-1}p^{q-1} + \cdots + g_1 p + 1$$

Multiply both the sides of this polynomial by p^i i.e.,

$$\boxed{p^i G(p) = p^{i+q} + g_{q-1}p^{i+q-1} + \cdots + g_1 p^{i+1} + p^i}$$

The above equation gives the polynomials for the rows of generating polynomials.

Systematic Form of Generator Matrix

The systematic form of generator matrix is given by,

$$G = [I_k : P_{k\times q}]_{k\times n}$$

The t^{th} row of this matrix will be represented in the polynomial form as,

$$\boxed{t^{th}\ row\ of\ G = p^{n-t} + R_t(p)} \qquad where\ t = 1,2,3, \ldots k$$

Let's divide p^{n-t} by a generator matrix $G(p)$. Then we can express the result of this division in terms of quotient and remainder. i.e.,

$$\frac{p^{n-t}}{G(p)} = Quotient + \frac{Remainder}{G(p)} \qquad --(1)$$

Here remainder will be a polynomial of degree less than *'q'*, since degree of $G(p)$ is *'q'*. The degree of quotient will depend upon value of *t*

Let's represent Remainder = $R_t(p)$

and Quotient = $Q_t(p)$

then equation (1) will be,

$$\frac{p^{n-t}}{G(p)} = Q_t(p) + \frac{R_t(p)}{G(p)}$$

$$i.e. \quad p^{n-t} = Q_t(p)G(p) \oplus R_t(p) \quad and \quad t = 1,2, \ldots t$$

We know that if $z = y \oplus t$, then $z \oplus y = t$ or $z \oplus t = y$. That is mod-2 addition and subtraction yields same results. Then we can write above equation as,

$$p^{n-t} \oplus R_t(p) = Q_t(p)G(p)$$

The above equation represents t^{th} row of systematic generator matrix.

11.(a). Explain the Encoding using an (n - k) Bit Shift Register in cyclic codes.

Or

(b).Explain the encoding scheme in Cyclic Code.

Figure shows the block diagram of a generalized (n, k) cyclic code. The symbols used to draw encoders are shown in Figure

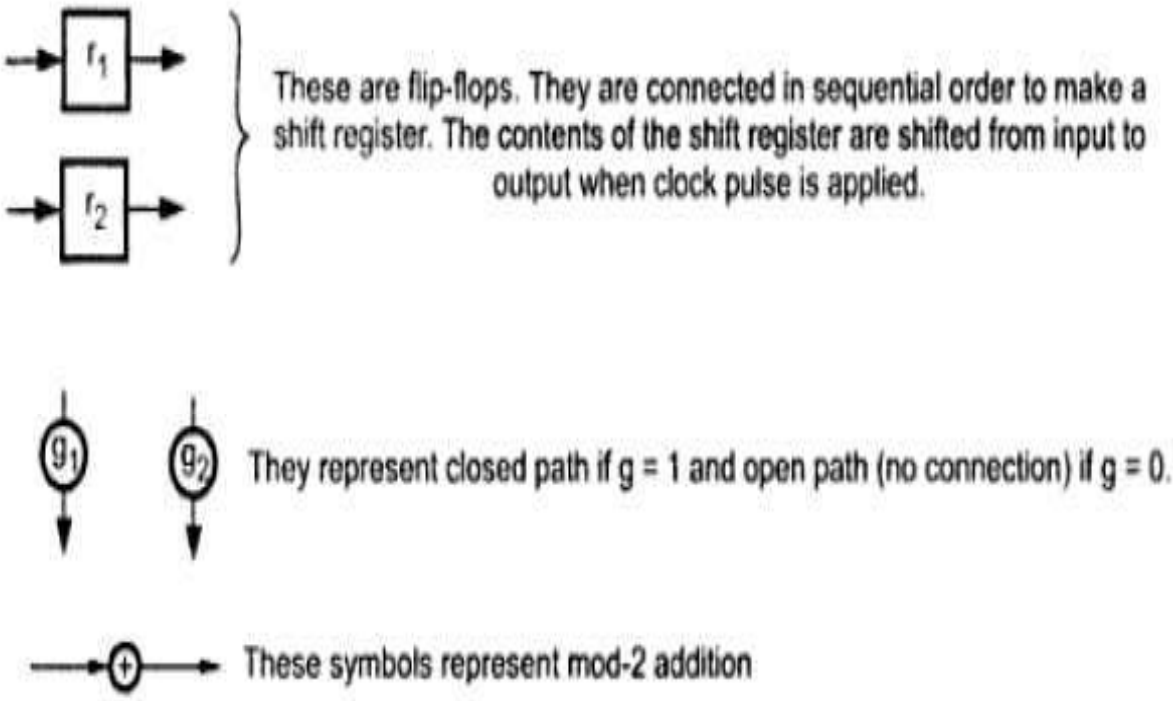


Figure: Various symbols used in encoder

Operation: The feedback switch is first closed. The output switch is connected to message input. All the shift registers are initialized to all zero state. The *k* message bits are shifted to the transmitter as well as shifted into the registers.

After the shift of *.k′* message bits the registers contain ' *q′* check bits. The feedback switch is now opened and output switch is connected to check bits position. With the every shift, the check bits are then shifted to the transmitter.

Here we observe that the block diagram performs the division operation and generates the remainder (i.e. check bits). This remainder is stored in the shift register after all message bits are shifted out.

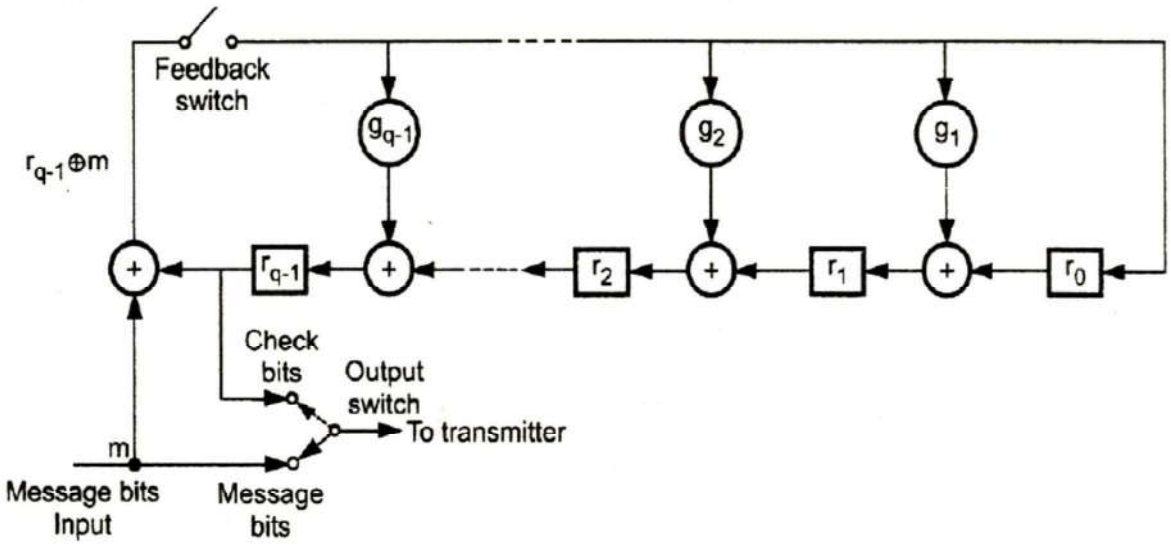


Fig 5.7 Encoder for Systematic (n,k) cyclic code

12.(a).Explain about Syndrome Decoding, Error Detection and Error Correction in cyclic codes.

Or

(b).Briefly Explain Syndrome Decoding.

In cyclic codes also during transmission some errors may occur. Syndrome decoding can be used to correct those errors. Let's represent the received code vector by Y. If 'E' represents an error vector then the correct code vector can be obtained as,

$$X = Y \oplus E$$

or we can write the above equation as,

$$Y = X \oplus E$$

We can write the above equation since it is rnod-2addition.

In the polynomial form we can write the above equation as,

$$Y(p) = X(p) + E(p)$$

$$since, \quad X(p) = M(p)G(p) \quad the\ above\ equation\ will\ be,$$

$$Y(p) = M(p)G(p) + E(p) \qquad --(1)$$

Let the received polynomial $Y(p)$ be divided by $G(p)$ i.e.

$$\frac{Y(p)}{G(p)} = Quotient + \frac{Remainder}{G(p)} \qquad --(2)$$

In the above equation if $Y(p) = X(p)$ i.e. if it does not contain any error then,

$$\frac{X(p)}{G(p)} = Quotient + \frac{Remainder}{G(p)}$$

Since $X(p) = M(p)G(p)$, quotient will be equal to $M(p)$ and remainder will be zero. This shows that if there is no error, then remainder will be zero. Here $G(p)$ is factor of code vector polynomial. Let's represent quotient by $Q(p)$ and remainder by $R(p)$ then equation (2) becomes,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)}$$

Clearly $R(p)$ will be the polynomial of degree less than or equal to $q - 1$. Multiply both sides of above equation by G *(p)* i.e,

$$Y(p) = Q(p)G(p) + R(p) \qquad --(3)$$

On comparing equation (1) and above equation (3) we obtain,

$$M(p)G(p) \oplus E(p) = E(p)G(p) \oplus R(p)$$
$$\therefore \quad E(p) = M(p)G(p) \oplus Q(p)G(p) \oplus R(p)$$

The above equation has all mod-2 additions. Therefore subtraction and addition is same.

$$\therefore \quad E(p) = [M(p) + Q(p)]G(p) + R(p)$$

This equation shows that for a fixed message vector and generator polynomial, an error pattern or error vector' *E* depends on remainder R. For every remainder' R' there will be specific error vector. Therefore we can call the remainder vector *'R'* as syndrome vector '5', or R *(p)* =5 *(p).* Therefore equation (3.3.59) will be,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{S(p)}{G(p)}$$

Thus the syndrome vector is obtained by dividing received vector $Y(p)$ by $G(p)$, i.e,

$$\boxed{S(p) = rem\left[\frac{Y(p)}{G(p)}\right]}$$

Block Diagram of Syndrome Calculator

Figure shows the generalized block diagram of a syndrome calculator.

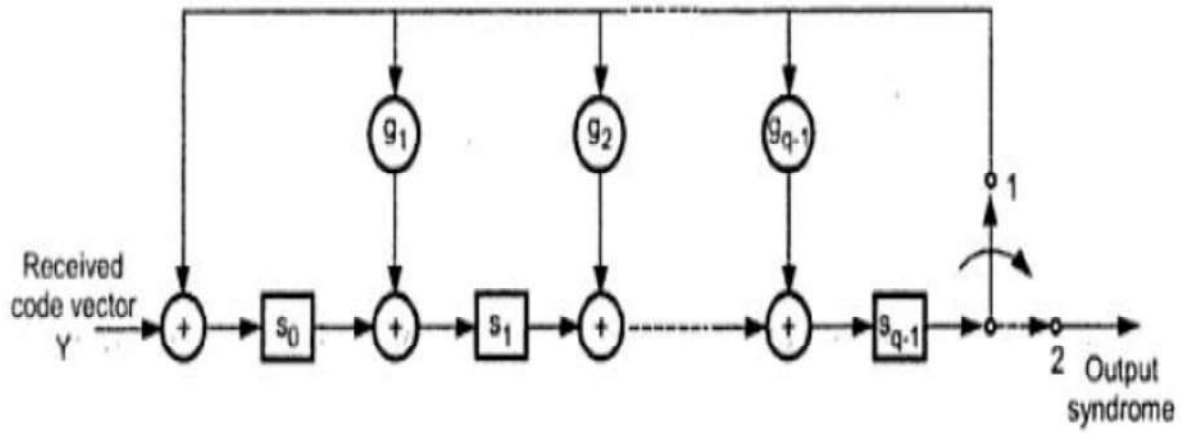


Figure:Computation of syndrome for an (n, k) cyclic code

In above figure there are 'q' stage shift register to generate' q' bit syndrome vector.

Operation: Initially all the shift register contents are zero and the switch is closed in position 1. The received vector Y is shifted bit by bit into the shift register. The contents of flip flops keep on changing according to input bits of Y and values of g_1, g_2 etc. After all the bits of Y are shifted, the 'q' flip-flops of shift register contains the q - bit syndrome vector. The switch is then closed to position 2 and clocks are applied to the shift register. The output is a syndrome vector $S = (s_{q-1}, s_{q-2}, \ldots s_1, s_0)$

13.(a) Explain about Decoder for Cyclic Codes

Or

(b). With block diagram,explain the decoder in cyclic codes.

Once the syndrome is calculated, then an error pattern is detected for that particular syndrome. When this error vector is added to the received vector Y, then it gives corrected code vector at the output. This decoding operation can be performed by the scheme shown in Figure.

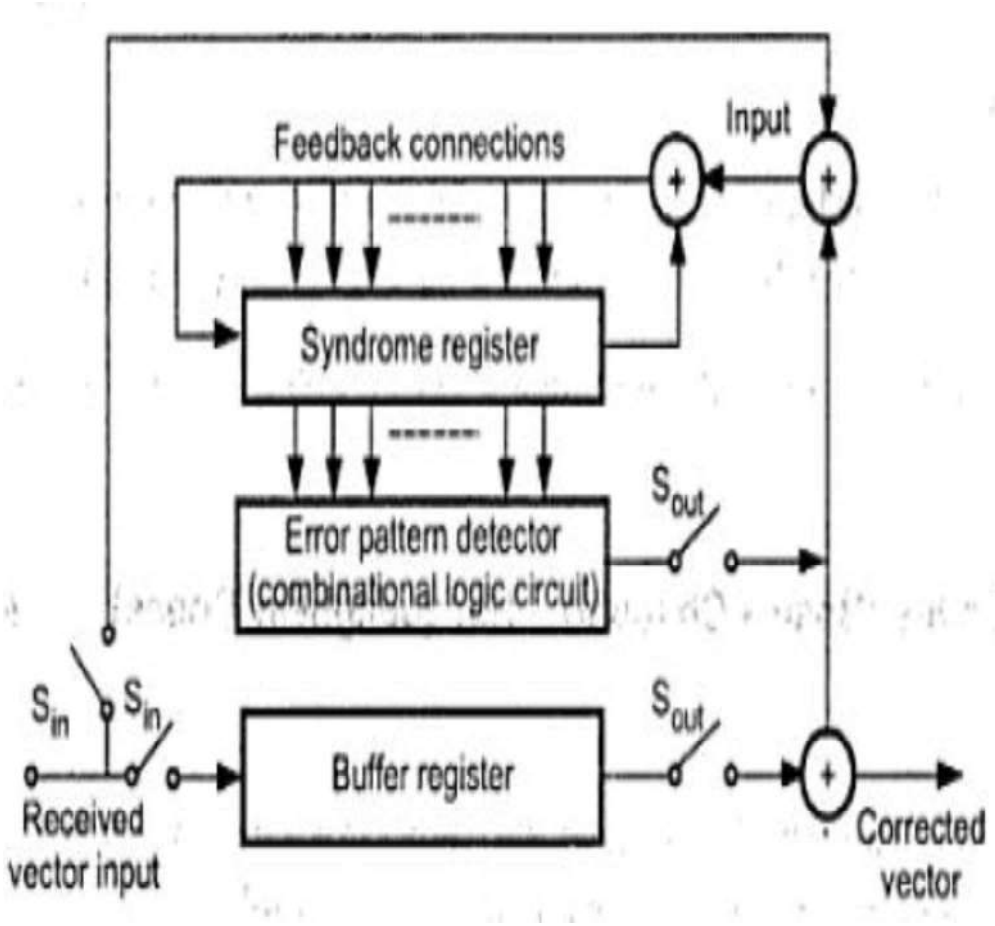


Figure: Generalized block diagram of decoder for cyclic codes

Operation of the decoder

The switches named S_{out} are opened and S_{in} are closed. The bits of the received vector Y are shifted into the buffer register as well as they are shifted into the syndrome calculator. When all the 'n' bits of the received vector Y are shifted into buffer register and syndrome calculator the syndrome register holds a syndrome vector. The syndrome vector is given to the error pattern detector. A particular syndrome detects a specific error pattern. The switches S_{in} are opened and S_{out} are closed. The shifts are then applied to the flip-flops of buffers register, error register (which holds error pattern) and syndrome register. The error pattern is then added bit by bit to the received vector (which is stored in buffer register). The output is the **corrected error free vector.**

14.(a) Explain in detail about convolutional codes

(or)

(b) Explain about 1/3 rate convolutional encoder.

Definition of Convolutional Coding

A convolutional coding is done by combining the fixed number of input bits. The input bits are stored in the fixed length shift register and they are combined with the help of mod-2 adders. This operation is equivalent to binary convolution and hence it is called **convolutional coding**. This concept is illustrated with the help of simple example given below.

Figure shows a convolutional encoder.

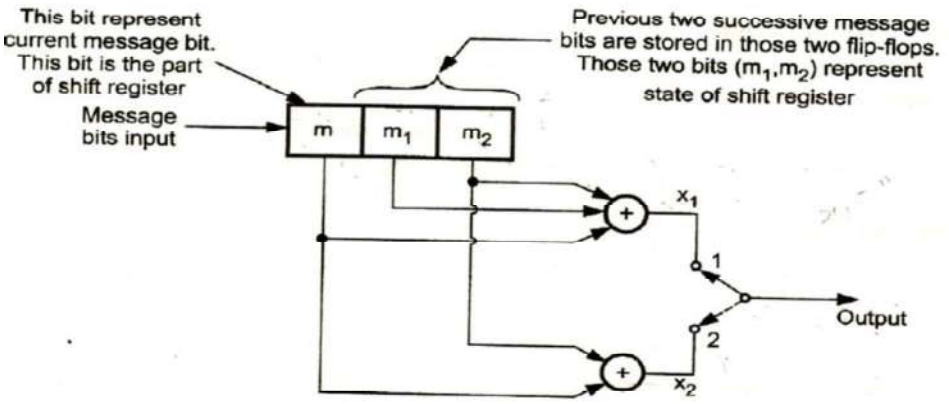


Figure:Convolutional encoder with k = 3, k =1 and n = 2

Operation:

Whenever the message bit is shifted to position *'m'*, the new values of x_1 *and* x_2 are generated depending upon m, m_1 *and* m_2. m_1 *and* m_2 store the previous two message bits. The current bit is present in *m*. Thus we can write,

$$x_1 = m_1 \oplus m_1 \oplus m_2 \qquad --(1)$$

$$and \qquad x_2 = m \oplus m_1 \qquad --(2)$$

The output switch first samples x_1 *and then* x_2. The shift register then shifts contents of m_1 *to* $m2$ and contents of m *to* m_1 Next input bit is then taken and stored in m. Again x_1 *and* x_2 are generated according to this new combination of m, m_1 *and* m_2 (equation (1) and equation (2)). The output switch then samples x_1 *then* x_2. Thus the output bit stream for successive input bits will be,

$$X = x_1 x_2 x_1 x_2 x_1 x_2 \ldots \ and \ so \ on$$

Here note that for every input message bit two encoded output bits x_1 *and* x_2 are transmitted. In other words, for a single message bit, the encoded code word is two bits i.e. for this convolutional encoder,

Number of message bits, k. = 1

Number of encoded output bits for one message bit, n = 2

Code Rate of Convolutional Encoder

The code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2}$$

In the encoder of Fig.5.10, observe that whenever a particular message bit enters n shift register, it remains in the shift register for three shifts i.e.,

First shift → Message bit is entered in position 'm'.

Second shift → Message bit is shifted In position $'m_1'$

Third shift → Message bit is shifted in position $'m_2'$

And at the fourth shift the message bit is discarded or simply lost by overwriting. We know that x_1 and x_2 are combinations of m_1, m_2, m_3. Since a single message bit remains in m during first shift, in m_1 during second shift and in m_2 during third shift; it influences output x_1 and x_2 for 'three' successive shifts.

Constraint Length (K):

The constraint length of a convolution code is defined as the number of shifts over which a single message bit can influence-the encoder output. It is expressed in terms of message bits. . .' .

For the encoder of Fig. 5.10 constraint length K = 3 bits. This is because in this encoder, a single message bit influences encoder output for three successive shifts. At the fourth shift, the message bit is lost and it has no effect on the output.

Dimension of the Code

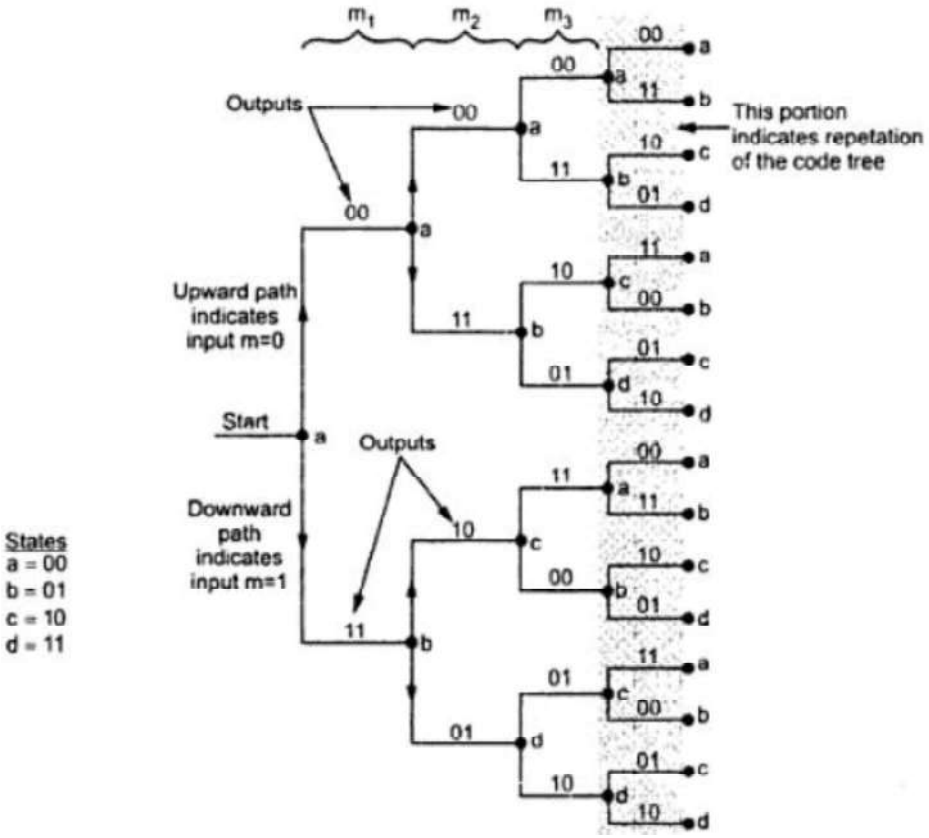
The dimension of the code is given by n and k, We know that 'k' is the number of message bits taken at a time by the encoder. And 'n' is the encoded output bits for one message bits. Hence the dimension of the code is (n, k). And such encoder is called. (n, k) convolutional encoder, For example, the encoder of Figure has the dimension of (2, 1).

code tree for convolutional encoder

Figure shows the code tree for this encoder. The code tree starts at node 'a'. If input message bit is '1' then path of the tree goes down towards node 'b' and output is 11. Otherwise if the input is m = 0 at node 'a', then path of the tree goes upward towards node 'a' and output is 00. Similarly, depending upon the input message bit, the path of the tree goes upward or downward. The nodes are marked with their states a, b, c or d. On the path between two nodes the outputs are shown. We have verified the part of this code tree for first three message bits as 110.

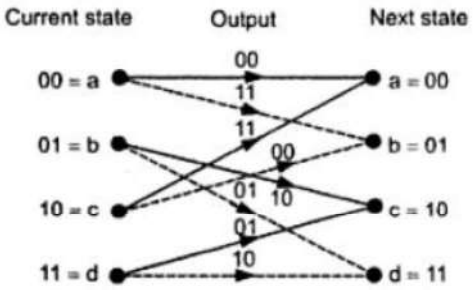
In the code tree of Figure, we find that the branch pattern begins to repeat after third bit. This is shown in figure. The repetition starts after 3rd bit, since

particular message bit is stored in the shift registers of the encoder for three shifts. If the length of the shift register is increased by one bit, then the pattern of code tree will repeat after fourth message bit.



Code Trellis (Represents Steady State Transitions)

Code trellis is the more compact representation of the code tree. We know that in the code tree there are four states (or nodes). Every state goes to some other state depending upon the input code. Trellis represents the single and unique diagram for such transitions. Fig. 5.16 shows code trellis diagram.



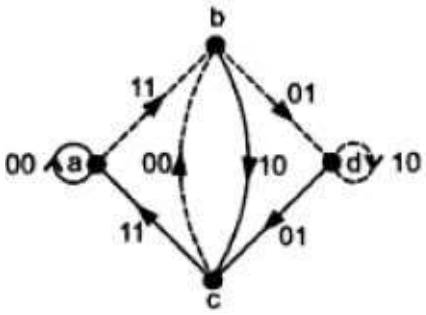
Code trellis of convolutional encoder

The nodes on the left denote four possible current states and those on the right represent next state. The solid transition line represents input m = 0 and broken

tine represents input $m = 1$. Along with each transition line the output x_1x_2 is represented during that transition. For example let the encoder be in current state of 'a'. If input m = 0, then next state will be 'a' with outputs $x_1x_2 = 11$. Thus code trellis is the compact representation of code tree.

State Diagram

If we combine the current and next states, then we obtain state diagram



State diagram for convolutional encoder

For example consider that the encoder is in state *'a'*. If input $m = 0$, then next state is same i.e. a (i.e. 00) with outputs x_1x_2= 00. This is shown by self loop at node 'a' in the state diagram. If input $m = 1$, then state diagram shows that next state is *'b'* with outputs x_1x_2= 11 .

15.(a)Explain in detail about convolutional codes

(or)

(b)Explain about 1/3 rate convolutional encoder

Let the sequence $\left\{g_0^{(1)}, g_1^{(1)}, g_2^{(1)}, \ldots\ g_m^{(1)}\right\}$, denote the impulse response of the adder which generates x_1 .an F'ig. 5.10 Similarly, Let the sequence $\left\{g_0^{(2)}, g_1^{(2)}, g_2^{(2)}, \ldots\ g_m^{(2)}\right\}$, denote the impulse response of the adder which generates x_2 in Fig. 5.10. These impulse responses are also called **generator sequences** of the code.

Let the incoming message sequence be m_0, m_1, m_2 .... The encoder generates the two output sequences x_1 and x_2 These are obtained by convolving the generator sequences with the message sequence. Hence the name convolutional code is given.

The sequence x_1 is given as,

$$x_1 = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \qquad i = 0,1,2,\ldots \qquad --(1)$$

Here m_{i-l}= 0 for all $l > i$. Similarly the sequence x_2 is given as,

$$x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l} \qquad i = 0,1,2,\ldots \qquad --(2)$$

Note: All additions in above equations are as per mod-2 addition rules.

As shown in the Figure the two sequences x_1 *and* x_2 are multiplexed by the switch. Hence the output sequence is given as,

$$\{x_i\} = \left\{x_0^{(1)} x_0^{(1)} x_1^{(1)} x_1^{(2)} x_2^{(1)} x_2^{(2)} x_3^{(1)} x_3^{(2)} \ldots\ldots\right\} \qquad --(3)$$

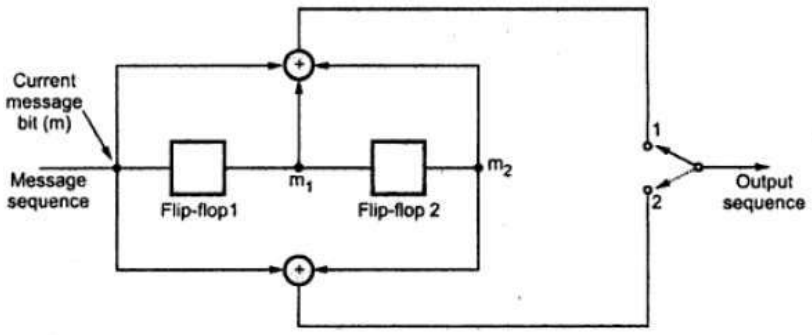
$$here, \quad v_1 = x_i^{(1)} = \left\{x_0^{(1)} x_1^{(1)} x_2^{(1)} x_3^{(1)} \ldots\right\}$$

$$and \quad v_2 = x_i^{(2)} = \left\{x_0^{(2)} x_1^{(2)} x_2^{(2)} x_3^{(2)} \ldots\right\}$$

Observe that bits from above two sequences are multiplexed in equation (4.4.8). The sequence $\{x_i\}$ is the output of the convolutional encoder

For the convolutional encoder of Figure determine the following:

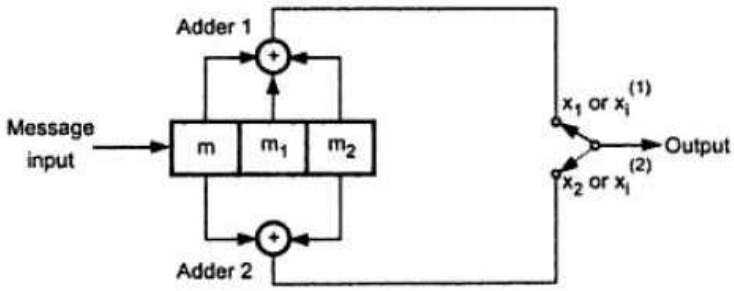
i. Dimension of tire code
ii. Code rate
iii. Constraint length
iv. Generating sequences (impulse responses)
v. Output sequence for message sequence of m = {1 0 0 1 1}



Convolutional encoder

Solution:

In the Figure observe that input of flip-flop 1 is the current message bit (m). The output of flip-flop 1 is the previous message bit i.e. m_1. The output of flip flop 2 is previous to previous message bit i.e. m_2. Hence above diagram can be redrawn as shown in Figure



Convolutional encoder of above figure redrawn alternately

i) Dimension of the code

Observe that encoder takes one message bit at a time. Hence k = 1. It generates two bits for every message bit. Hence n = 2. Hence,

Dimension = (n, k) = (2, 1)

ii) Code rate

Code rate is given as,

$$r = \frac{k}{n} = \frac{1}{2}$$

iii) Constraint length

Here note that every message bit, affects output bits for three successive shifts. Hence,

Constraint length K = 3 bits

iv) Generating sequence.

In Fig. 5.12 observe that x_1 i.e. $x_i^{(1)}$ is generated by adding all the three bits. Hence generating sequence $g_{(i)}^1$ is given as

$$g_i^{(1)} = [1\ 1\ 1]$$

$$here, \quad g_0^{(1)} = 1\ represents\ connection\ of\ bit\ m$$

$$g_1^{(1)} = 1\ represents\ connection\ of\ bit\ m_1$$

$$g_2^{(1)} = 1 \; represents \; connection \; of \; bit \; m_2$$

x_2 is generated by addition of first and last bits. Hence its generating sequence is given as,

$$g_i^{(2)} = [1\ 0\ 1]$$

$$here, \qquad g_0^{(2)} = 1 \; represents \; connection \; of \; bit \; m$$

$$g_1^{(2)} = 0 \; represents \; that \; m_1 \; is \; not \; connected$$

$$g_2^{(2)} = 1 \; represents \; connection \; of \; bit \; m_2$$

The above sequences are also called impulse responses.

v) To obtain output sequence

The given message sequence is,

$$m = (m_0\ m_1\ m_2\ m_3\ m_4) = (1\ 0\ 0\ 1\ 1)$$

To obtain output due* to *adder 1

Then from equation (1) we can write,

$$x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)}\, m_{i-l} \qquad --(4)$$

with $i = 0$ above equation becomes,

$$\therefore \quad x_0^{(1)} = \sum_{l=0}^{M} g_l^{(1)}\, m_{i-l}$$

$$\therefore \quad x_0^{(1)} = g_0^{(1)} m_0$$

$$= 1 \times 1 = 1, \qquad Here \;\; g_0^{(1)} = 1 \;\; and \;\; m_0 = 1$$

i=1 in eqn (4)

$$x_1^{(1)} = g_0^{(1)} m_1 \oplus g_1^{(1)} m_0$$

$$= (1 \times 0) \oplus (1 \times 1) = 1$$

Here note that additions are mod-2 type.

i=2 in eqn (4)

$$x_2^{(1)} = g_0^{(1)} m_2 \oplus g_1^{(1)} m_1 \oplus g_2^{(1)} m_0$$

$$= (1 \times 0) \oplus (1 \times 0) \oplus (1 \times 1) = 1$$

i=3 in eqn (4)

$$x_3^{(1)} = g_0^{(1)} m_3 \oplus g_1^{(1)} m_2 \oplus g_2^{(1)} m_1$$

$$= (1 \times 1) \oplus (1 \times 0) \oplus (1 \times 0) = 1$$

i =4 in equation (4),

$$x_4^{(1)} = g_0^{(1)} m_4 \oplus g_1^{(1)} m_3 \oplus g_2^{(1)} m_2$$

$$= (1 \times 1) \oplus (1 \times 1) \oplus (1 \times 0) = 0$$

i =5 in equation (4),

$$x_5^{(1)} = g_0^{(1)} m_5 \oplus g_1^{(1)} m_4 \oplus g_2^{(1)} m_3$$

$$= g_1^{(1)} m_4 \oplus g_2^{(1)} m_3 \quad since\ m_5\ is\ not\ available$$

$$= (1 \times 1) \oplus (1 \times 1) = 0$$

i =6 in equation (4),

$$x_6^{(1)} = g_0^{(1)} m_6 \oplus g_1^{(1)} m_5 \oplus g_2^{(1)} m_4$$

$$= g_2^{(1)} m_4 \quad since\ m_6\ and\ m_5\ are\ not\ available$$

$$= (1 \times 1) = 1$$

Thus the output of adder 1 is,

$$\boxed{x_1 = x_i^{(1)} = \{1\ 1\ 1\ 1\ 0\ 0\}}$$

To obtain output due to adder 2

Similarly from equation (2),

$$x_1 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l}$$

$$and \quad m_{i-l} = 0 \quad for\ all\ l > i$$

With i = 0 in above eqn we get,

$$x_0^{(2)} = g_0^{(2)} m_0 = (1 \times 1) = 1 \qquad here\ g_0^{(2)} = 1\ and\ m_0 = 1$$

With i= 1

$$x_1^{(2)} = g_0^{(2)} m_1 \oplus g_1^{(1)} m_0$$
$$= (1 \times 0) \oplus (0 \times 1) = 0$$

With i = 2,

$$x_2^{(2)} = g_0^{(2)} m_2 \oplus g_1^{(2)} m_1 \oplus g_2^{(2)} m_0$$
$$= (1 \times 0) \oplus (0 \times 0) \oplus (1 \times 1) = 1$$

With; = 3,

$$x_3^{(2)} = g_0^{(2)} m_3 \oplus g_1^{(2)} m_2 \oplus g_2^{(2)} m_1$$
$$= (1 \times 1) \oplus (0 \times 0) \oplus (1 \times 0) = 1$$

With i=4,

$$x_4^{(2)} = g_0^{(2)} m_4 \oplus g_1^{(2)} m_3 \oplus g_2^{(2)} m_2$$
$$= (1 \times 1) \oplus (0 \times 1) \oplus (1 \times 0) = 1$$

With i=5,

$$x_5^{(2)} = g_1^{(2)} m_4 \oplus g_2^{(2)} m_3$$
$$= (0 \times 1) \oplus (1 \times 1) = 1$$

With i=6,

$$x_6^{(2)} = g_2^{(2)} m_4$$
$$= 1 \times 1 = 1$$

Thus the sequence x_2 is,

$$x_2 = x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

To obtain multiplexed sequence of x_1 and x_2 as per equation (3)

The two sequences x_1 and x_2 are multiplexed to get the final output i.e.

$$x_i = x_0^{(1)} x_0^{(2)} x_1^{(1)} x_1^{(2)} x_2^{(1)} x_2^{(2)} x_3^{(1)} x_3^{(2)} x_4^{(1)} x_4^{(2)} x_5^{(1)} x_5^{(2)} x_6^{(1)} x_6^{(2)}$$
$$= \{1\ 1, 1\ 0, 1\ 1, 1\ 1, 0\ 1, 0\ 1, 1\ 1\}$$

Transform Domain Approach of Convolutional Encoder

In the previous section we observed that the convolution of generating sequence and message sequence takes place. These calculations can be simplified by applying the transformations to the sequences. Let the impulse responses be represented by polynomials. i.e.

$$g^{(1)}(p) = g_0^{(1)} + g_1^{(1)}p + g_2^{(1)}p^2 + \cdots + g_M^{(1)}p^M$$

Similarly,

$$g^{(2)}(p) = g_0^{(2)} + g_1^{(2)}p + g_2^{(2)}p^2 + \cdots + g_M^{(2)}p^M$$

Thus the polynomials can be written for other generating sequences. The variable 'p' is unit delay operator in above equations. It represents the time delay of the bits in impulse response.

Simtlarly we can write the polynomial (or message polynomial i.e.

$$m(p) = m_0 + m_1p + m_2p^2 + \cdots + m_{L-1}p^{L-1}$$

Here L is the length of the message sequence. The convolution sums are converted to polynomial multiplications in the transform domain. i.e.,

$$\boxed{\begin{aligned} x^{(1)}(p) &= g^{(1)}(p).m(p) \\ x^{(2)}(p) &= g^{(2)}(p).m(p) \end{aligned}} \qquad --(5)$$

The above equations are the output polynomials of sequences $x_i^{(1)}$ *and* $x_i^{(2)}$.

Note : All additions in above equations are as per mod 2 addition rules.

Solution:

a) To obtain generating polynomial for adder-1 :

The first generating sequence is given by,

$$g_i^{(1)} = \{1\ 1\ 1\}$$

Hence its polynomial can be obtained as follows:

$$g^{(1)}(p) = 1 + 1 \times p + 1 \times p^2$$

$$= 1 + p + p^2$$

b) To obtain generating polynomial for adder-2

The second generating sequence is given by,

$$g_i^{(2)} = \{1\ 0\ 1\}$$

Hence its polynomial can be obtained as follows:

$$g^{(2)}(p) = 1 + 0 \times p + 1 \times p^2$$

$$= 1 + p^2$$

c) To obtain message polynomial

The message sequence is,

$$m = (1\ 0\ 0\ 1\ 1)$$

Hence its polynomial can be obtained as,

$$m(p) = 1 + 0 \times p + 0 \times p^2 + 1 \times p^3 + 1 \times p^4$$
$$= 1 + p^3 + p^4$$

d) To determine the output due to adder-1

Now $x^{(1)}(p)$ can be obtained from equation (5). i.e.

$$x^{(1)}(p) = g^{(1)}(p).m(p)$$
$$= (1 + p + p^2)(1 + p^3 + p^4)$$
$$= 1 + p + p^2 + p^3 + p^6$$

The above polynomial can also be written as,

$$x^{(1)}(p) = 1 + (1 \times p) + (1 \times p^2) + (1 \times p^3) + (0 \times p^4) + (0 \times p^5) + (1 \times p^6)$$

Thus the output sequence $x_i^{(1)}$ is,

$$x_i^{(1)} = \{1\ 1\ 1\ 1\ 0\ 0\ 1\}$$

e) To determine the output due to adder·2

Similarly polynomial ,$x^{(2)}(p)$ can be obtained as,

$$x^{(2)}(p) = g^{(2)}(p).m(p)$$
$$= (1 + p^2)(1 + p^3 + p^4)$$
$$= 1 + p^2 + p^3 + p^4 + p^5 + p^6$$

Thus the output sequence $x_i^{(2)}(p)$ is,

$$x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

f) To determine the multiplexed output sequence

The multiplexed output sequence will be as follows:

$$\{x_i\} = \{1\ 1, 1\ 0, 1\ 1, 1\ 1, 0\ 1, 0\ 1, 1\ 1\}$$

Here note that very few calculations are involved in transform domain.

16.(a). Explain the Decoding Methods of Convolutional Codes

Or

(b).Explain Viterbi algorithm for decoding of Convolutional codes.

These methods are used for decoding of convolutional codes. They are viterbi algorithm, sequential decoding and feedback decoding.

Viterbi Algorithm for Decoding of Convolutional Codes (Maximum Likelihood Decoding)

Let's represent the received signal by Y. Convolutional encoding operates continuously on input data. Hence there are no code vectors and blocks. Let's assume that the transmission error probability of symbols 1's and 0's is same. Let's define an integer variable metric as follows.

Metric:

It is the discrepancy between the received signal Y and the decoded signal at particular node. This metric can be added over few nodes for a particular path.

Surviving Path:

This is the path of the decoded signal with minimum metric.

In viterbi decoding a metric is assigned to each surviving path. (Metric of a particular path is obtained by adding individual metric on the nodes along that path). Y is decoded as the surviving path with smallest metric.

Consider the following example of viterbi decoding. Let the signal being received is encoded by the encoder of Fig. 5.10. Let the first six received bits be

$$Y = 11\ \ 01\ \ 11$$

***a) Decoding of first message bit for Y* = 11**

Note that for single bit input the encoder transmits two bits $(x_1 x_2)$ outputs. These outputs are received at the decoder and represented by Y. Thus Y given above represents the outputs for three successive message bits. Assume that the decoder is at state a_0. Now look at the code trellis diagram of Figure for this encoder. It shows that if the current state is 'a', then next state will be 'a' or 'b'. This is shown in Figure. Two branches are shown from a_0. One branch is at next node a_1 representing decoded signal as 00 and other branch is at b_1 representing decoded signal as 11.

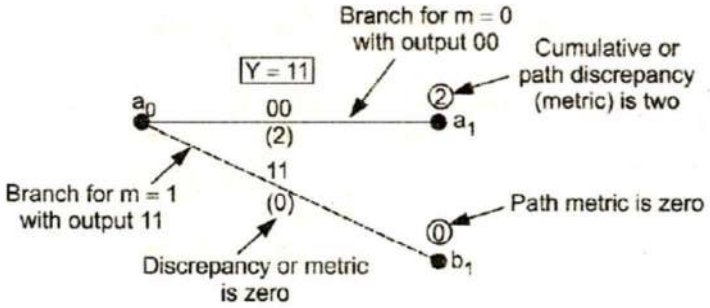


Figure: Viterbi decoder results for first message bit

The branch from $a_0 b_1$ to represents decoded output as 11 which, is same as received signal at that node i.e. 11. Thus there is no discrepancy between received signal and decoded signal. Hence 'Metric' of that branch is zero. This metric is shown in brackets along that branch. The metric of branch from a_0 to a_1 is two. The encoded number near a node shows path metric reaching to the node.

b) Decoding of second message bit for Y = 01

When the next part of bits $Y = 01$ is received at nodes a_1 and b_1, then from nodes a_1 and b_1 four possible next states a_2, b_2, c_2 *and* d_2 are possible. Figure shows all these branches, their decoded outputs and branch metrics corresponding to those decoded outputs. The encircled number near a_2, b_2, c_2 *and* d_2 indicate path metric emerging from a_0. For example the path metric of path a_0, a_1, a_2 is 'three'. The path metric of path $a_0 b_1 d_2$ is zero

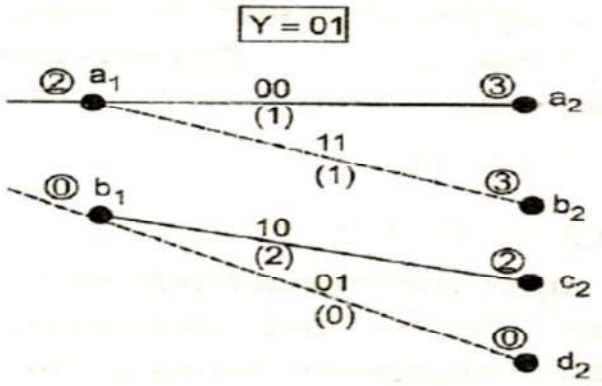


Figure: Viterbi decoder results for second message bit

c) *Decoding of 3rd message bit for* Y = 11

Figure shows the trellis diagram for all the six bits of Y.

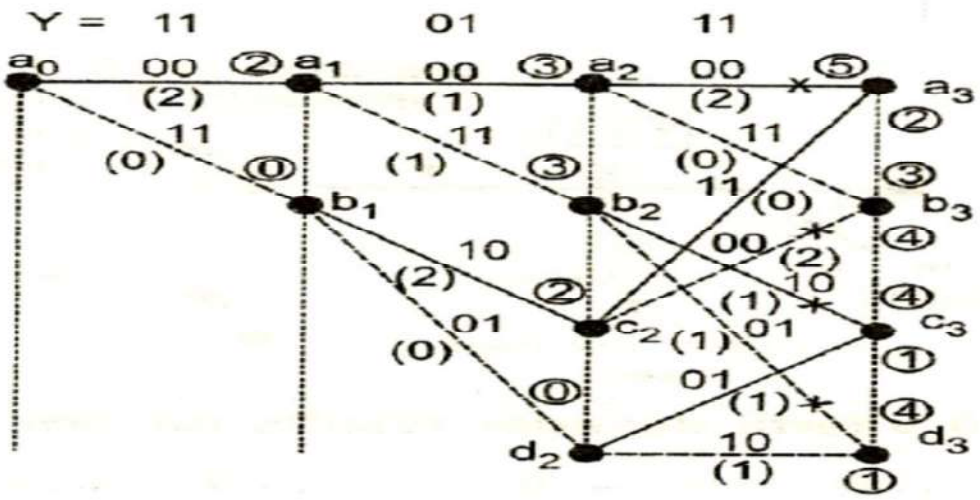


Figure: Paths and their metrics for viterbi decoding

Figure shows the nodes with their path metrics on the right hand side at the end of sixth bit of Y. Thus two paths are common to node *'a'*. One path is $a_0a_1a_2a_3$ with metric 5. The other path· is $a_0b_1c_2a_3$ with metric 2. Similarly there are two paths at other nodes also. According to viterbi decoding, only one path with lower metric should be retained at particular node. The paths marked with x (cross) are cancelled because they have higher metrics than other path coming to that particular node. These four paths with lower metrics are stored in the decoder and the decoding continues to next received bits.

d) Further explanation of viterbi decoding for* 12 *message bits

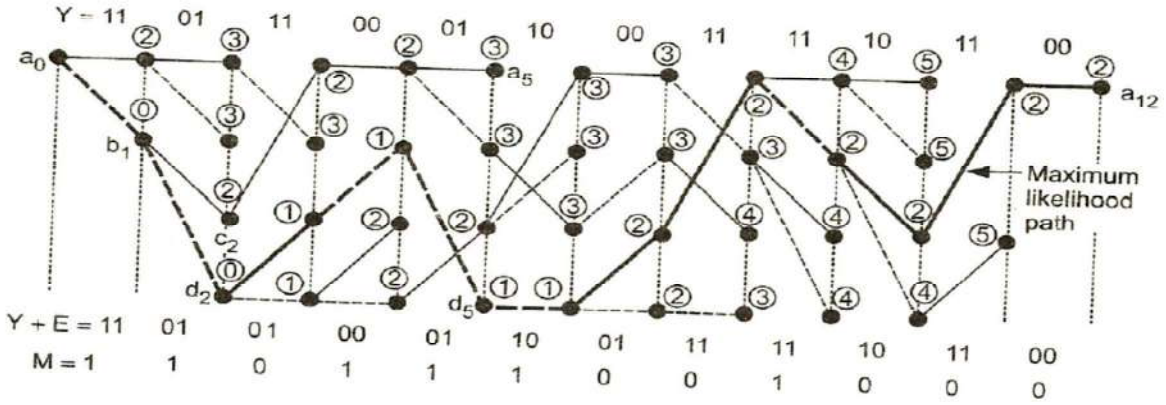


Figure Viterbi decoding

Figure shows the continuation of Figure for a message 12 bits. Observe that in this figure, received bits Y are marked at the top of the decoded value of output i.e. Y+E is marked at the bottom and decoded message signal is also marked

Only one path of particular node is kept which is having lower metric. In case if there are two paths having same metric, then anyone of them is continued.

Observe that a node $'a_{12}'$ only one path arrives with metric two. This path is shown by a thick line. Since this path is lowest metric it is the surviving path and hence Y is decoded from this path. All the decoded values of output are taken from the outputs of this path. Whenever this path is broken it shows message bit m = 1 and if it is continuous, message bit m =0 between two nodes.

The method of decoding used in viterbi decoding is called **maximum likelihood decoding**.

e) Surviving paths

During decoding we find that a viterbi decoder has to store four surviving paths for four nodes.

$$Surviving\ paths\ =\ 2^{(K-1)k}$$

Here K is constraint length and k is number of message bits.

For the encoder for Fig. 5.10 K = 3 and k = 1

$$\therefore Surviving\ paths\ =\ 2^{(3-1)\,x\,1}\ =\ 4$$

Thus the viterbi decoder has to store four surviving paths always. If the number of message bits to be decoded are very large, then storage requirement is also large since the decoder has to store multiple paths. To avoid this problem metric diversion effect is used.

f) Metric Diversion Effect:

For the two surviving paths originating from the same node, the running metric of less likely path tends to increase more rapidly than the metric of other path within about $5(k\ -1)$ branches from the common node. This is called **metric divergence effect**. For example consider the two paths coming from node b_1. One path comes at a_5 and other path comes at d_5. The path at a_5 is less likely and hence its metric is more compared to the path' at d_5. Hence at node d_5 only the surviver path is selected and the message bits are decoded. The fresh paths are started from d_5. Because of this, the memory storage is reduced since complete path need not be stored.

25. For the convolutional encoder arrangement shown in Figure, draw the state diagram and hence trellis diagram. Determine output digit sequence for the data digits 1 1 0 1 0 1 0 0. What are the dimensions of the code (n, k) and constraint length ? Use viterbi's algorithm to decode the sequence, 100 110 111 101 001 101 001 010.

Solution:

i) To obtain dimension of the code:

Observe that one message bit is taken at a time in the encoder . Hence $k = 1$. There are three output bits for every message bit. Hence $n = 3$. Therefore the dimension of the code is $(n, k) = (3, 1)$.

ii) Constraint length

Here note that every message bit affects three output bits. Hence,

$$Constraint\ length\ K\ =\ 3\ bits.$$

iii) To obtain code trellis and state diagram

Figure shows the code trellis of the given encoder

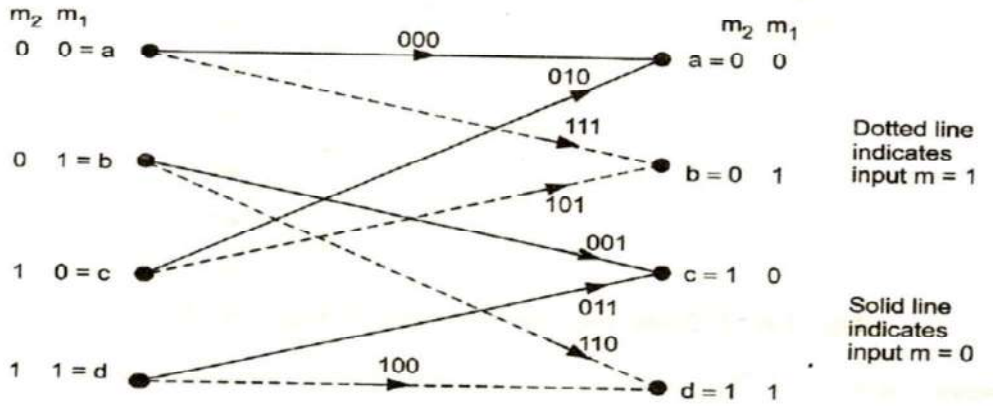
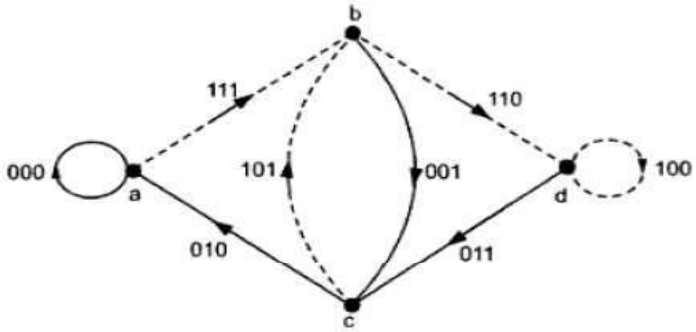


Figure Code trellis of encoder

The nodes in the code trellis can be combined to form state diagram as shown in Figure



State diagram of the encoder

iv)To determine output sequence

a) Determine generator polynomials

The generating sequence can be written for $x_i^{(1)}$ from Fig. 5.22 as,

$g_i^{(1)} = \{1, 0, 0\}$ *since only m is connected*

Similarly generating sequence for $x_i^{(2)}$ will be,

$g_i^{(2)} = \{1, 0, 1\}$ *since m and m_2 are connected*

And generating sequence for , $x_i^{(3)}$ will be,

$g_i^{(3)} = \{1, 1, 0\}$ *since m and m_1 are connected*

Hence the corresponding generating polynomials can be written as,

$$g^{(1)}(p) = 1$$

$$g^{(2)}(p) = 1 + p^2$$

$$g^{(3)}(p) = 1 + p$$

b) Determine message polynomials

The given message sequence is,

$$m = \{1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\}$$

Hence the message polynomial will be,

$$m(p) = 1 + p + p^3 + p^5$$

c) Obtain output for $g_i^{(1)}$

The first sequence $x_i^{(1)}$ is given as,

$$x_i^{(1)} = g^{(1)}(p).m(p)$$

$$= 1(1 + p + p^3 + p^5)$$

$$= 1 + p + p^3 + p^5$$

Hence, the corresponding sequence will be,

$$\left\{x_i^{(1)}\right\} = \{1\ 1\ 0\ 1\ 0\ 1\}$$

d) *Obtain output for $g_i^{(2)}$*

The first sequence $x_i^{(2)}$ is given as,

$$x_i^{(2)} = g^{(2)}(p).m(p)$$

$$= (1 + p^2)(1 + p + p^3 + p^5)$$

$$= 1 + p + p^2 + p^7$$

Hence the corresponding sequence will be,

$$x_i^{(2)} = \{1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\}$$

e) Obtain output for $g_i^{(3)}$

The third sequence $x_i^{(3)}$ is given as,

$$x_i^{(3)} = g^{(3)}(p).m(p)$$

$$= (1 + p)(1 + p + p^3 + p^5)$$

$$= 1 + p^2 + p^3 + p^4 + p^5 + p^6$$

Hence the corresponding sequence is,

$$x_i^{(3)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

f) To multiplex three output sequences

The three sequences $x_i^{(1)}, x_i^{(2)} and\ x_i^{(3)}$ are made equal in length i.e. 8 bits. Hence zeros are appended in sequence $x_i^{(1)}\ and\ x_i^{(2)}$. These sequences are shown below:

$$x_i^{(1)} = \{1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\}$$

$$x_i^{(2)} = \{1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\}$$

$$x_i^{(3)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\}$$

The bits from above three sequences are multiplexed to give the output sequence i.e.

$$\{x_i\} = \{111\ \ 110\ \ 011\ \ 101\ \ 001\ \ 101\ \ 001\ \ 010\}$$

v) Viterbi algorithm to decode the given sequence

Fig. 5.25 (next page) shows the diagram based on viterbi decoding. It shows received sequence at the top. The decoded (Y + E) sequence and decoded message sequence is shown at the bottom.

The dark line shows maximum likelihood path. It has the lowest running metric, i.e. 3. At any point only four paths are retained. The decoded message sequence is,

$$m = \{110\ 1\ 0\ 100\}$$

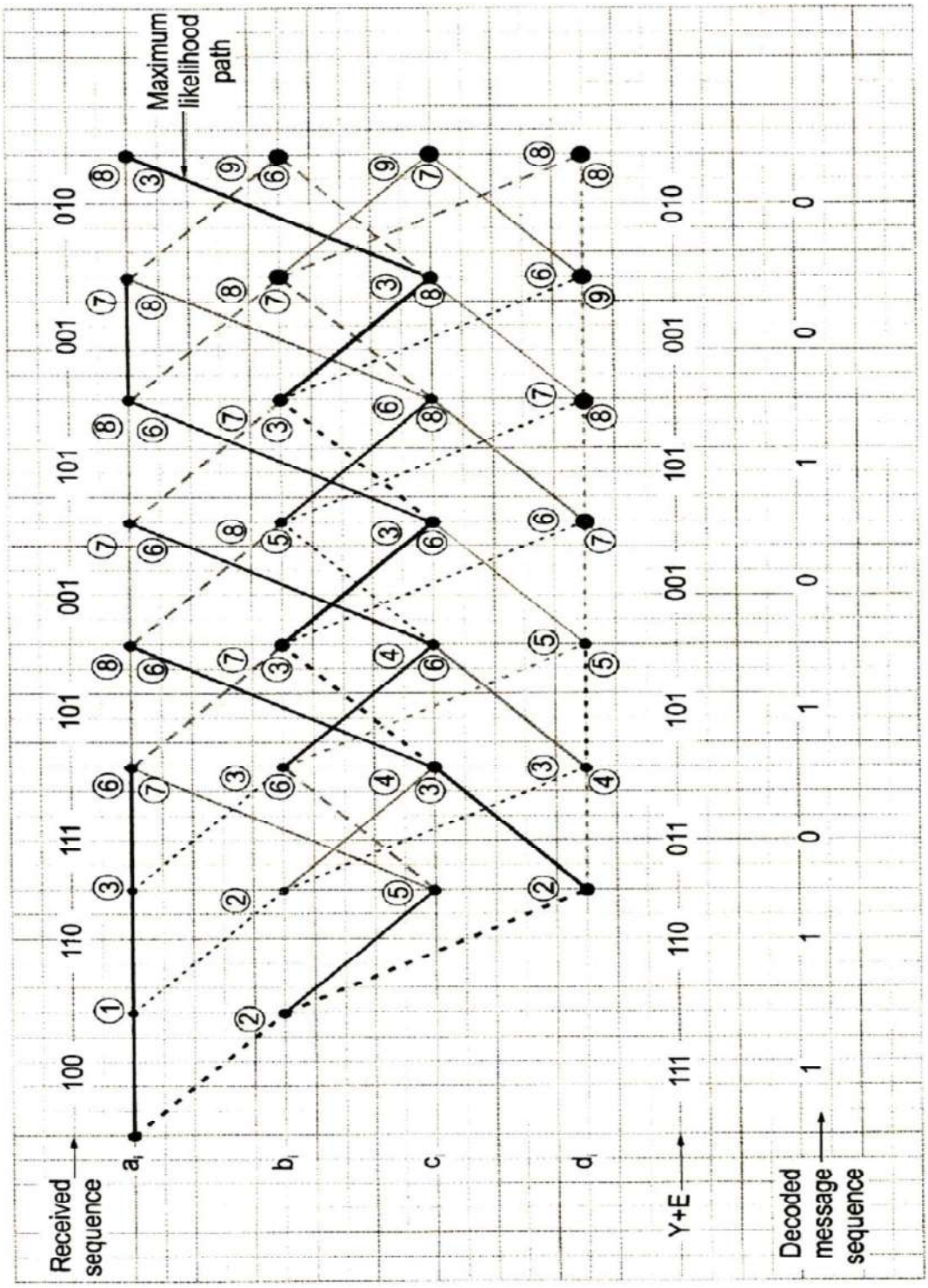


Figure Viterbi decoding