Information Theory and Coding

Subject Code : 10EC55	IA Marks : 25
No. of Lecture Hrs/Week : 04	Exam Hours : 03
Total no. of Lecture Hrs. : 52	Exam Marks : 100

PART - A

Unit – 1:

Information Theory: Introduction, Measure of information, Average information content of symbols in long independent sequences, Average information content of symbols in long dependent sequences. Mark-off statistical model for information source, Entropy and information rate of mark-off source. **6 Hours**

Unit – 2:

Source Coding: Encoding of the source output, Shannon's encoding algorithm. Communication Channels, Discrete communication channels, Continuous channels.

6 Hours

Unit – 3:

Fundamental Limits on Performance: Source coding theorem, Huffman coding, Discrete memory less Channels, Mutual information, Channel Capacity. **6 Hours**

Unit – 4:

Channel coding theorem, Differential entropy and mutual information for continuous ensembles, Channel capacity Theorem. **6 Hours**

PART - B

Unit – 5:

Introduction to Error Control Coding: Introduction, Types of errors, examples, Types of codes Linear Block Codes: Matrix description, Error detection and correction, Standard arrays and table look up for decoding. **7 Hours**

Unit – 6:

Binary Cycle Codes, Algebraic structures of cyclic codes, Encoding using an (n-k) bit shift register, Syndrome calculation. BCH codes. **7 Hours**

Unit – 7:

RS codes, Golay codes, Shortened cyclic codes, Burst error correcting codes. Burst and Random Error correcting codes. **7 Hours**

Unit – 8:

Convolution Codes, Time domain approach. Transform domain approach.

7Hours

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley,
1996. Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007
Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

INDEX SHEET

PART A

Unit – 1: Information Theory

Syllabus:

Introduction, Measure of information, Average information content of symbols in long independent sequences, Average information content of symbols in long dependent sequences. Mark-off statistical model for information source, Entropy and information rate of mark-off source. **6 Hours**

Text Books:

- Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996.

Reference Books:

- Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

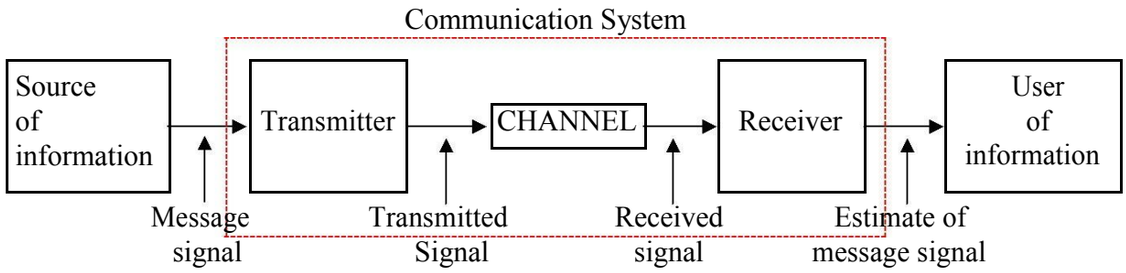
Unit – 1: Information Theory

1.1 Introduction:

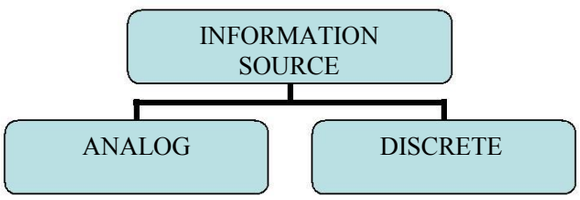
- **Communication**

Communication involves explicitly the transmission of information from one point to another, through a succession of processes.

- **Basic elements to every communication system**
  - Transmitter
  - Channel and
  - Receiver



- **Information sources are classified as:**



- **Source definition**

Analog : Emit a continuous – amplitude, continuous – time electrical wave from. Discrete : Emit a sequence of letters of symbols.

The output of a discrete information source is a string or sequence of symbols.

1.2 Measure the information:

To measure the information content of a message quantitatively, we are required to arrive at an intuitive concept of the amount of information.

Consider the following examples:

A trip to Mercara (Coorg) in the winter time during evening hours,

1. It is a cold day
2. It is a cloudy day
3. Possible snow flurries

Amount of information received is obviously different for these messages.

- Message (1) Contains very little information since the weather in coorg is 'cold' for most part of the time during winter season.
- The forecast of 'cloudy day' contains more informat ion, since it is not an event that occurs often.
- In contrast, the forecast of 'snow flurries' convey s even more information, since the occurrence of snow in coorg is a rare event.

On an intuitive basis, then with a knowledge of the occurrence of an event, what can be said about the amount of information conveyed?

It is related to the probability of occurrence of the event.

What do you conclude from the above example with regard to quantity of information?

Message associated with an event 'least likely to occur' contains most information. The

information content of a message can be expressed quantitatively as follows:

The above concepts can now be formed interns of probabilities as follows:

Say that, an information source emits one of 'q' po ssible messages $m_1, m_2 \ldots\ldots m_q$ with $p_1, p_2 \ldots\ldots p_q$ as their probs. of occurrence.

Based on the above intusion, the information content of the k^{th} message, can be written as

$$I(m_k) \ \alpha \ \frac{1}{p_k}$$

Also to satisfy the intuitive concept, of information.

$I(m_k)$ must zero as p_k 1

Therefore,

$I(m_k) > I(m_j)$; if $p_k < p_j$

$I(m_k)$ $O(m_j)$; if p_k 1 ------ I

$I(m_k) \geq O$; when $O < p_k < 1$

Another requirement is that when two independent messages are received, the total information content is –

Sum of the information conveyed by each of the messages.

Thus, we have

$$I(m_k \ \& \ m_q) \triangleq I(m_k \ \& \ m_q) = I_{mk} + I_{mq} \quad \text{------ I}$$

$\therefore$ We can define a measure of information as –

$$\frac{1}{p_k} \quad \text{------ III}$$

Unit of information measure

Base of the logarithm will determine the unit assigned to the information content.

Natural logarithm base : 'nat'

Base - 10 : Hartley / decit

Base - 2 : bit

Use of binary digit as the unit of information?

Is based on the fact that if two possible binary digits occur with equal proby ($p_1 = p_2 = ½$) then the correct identification of the binary digit conveys an amount of information.

$I(m_1) = I(m_2) = -\log_2(½) = 1$ bit

$\therefore$ One bit is the amount if information that we gain when one of two possible and equally likely events occurs.

Illustrative Example

A source puts out one of five possible messages during each message interval. The probs. of these messages are $p_1 = \frac{1}{2}$; $p_2 = \frac{1}{4}$; $p_1 = \frac{1}{4}$: $p_1 = \frac{1}{16}$, $p_5 \frac{1}{16}$

What is the information content of these messages?

$I(m_1) = -\log_2 \frac{1}{2} = 1$ bit

$I(m_2) = -\log_2 \frac{1}{4} = 2$ bits

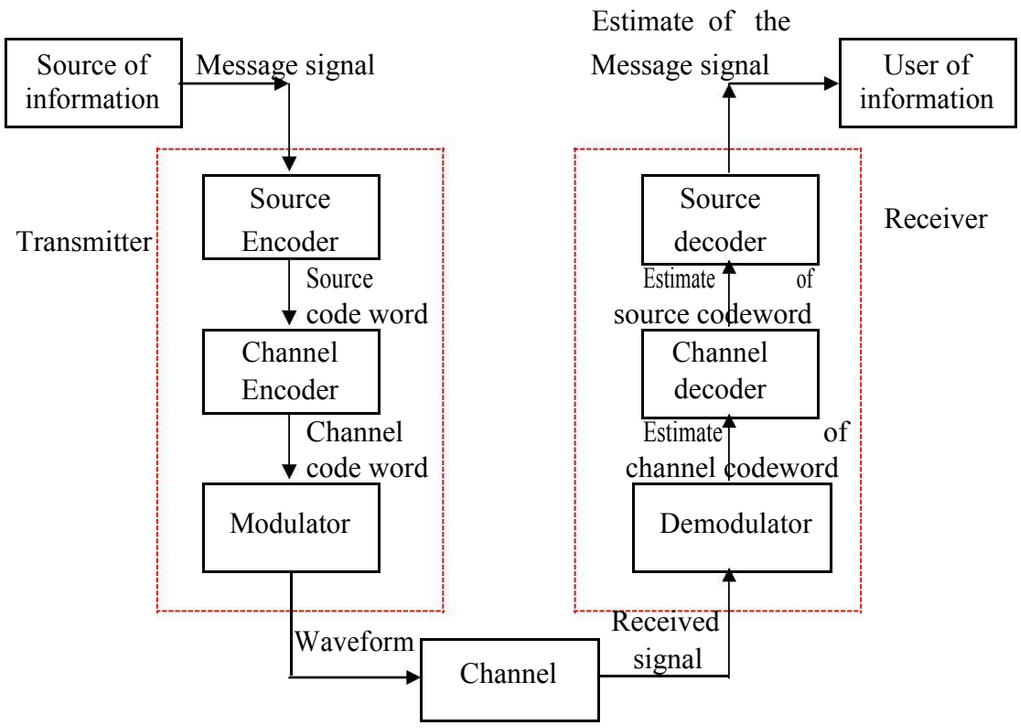
$I(m_3) = -\log \frac{1}{8} = 3$ bits

$I(m_4) = -\log_2 \frac{1}{16} = 4$ bits

$I(m_5) = -\log_2 \frac{1}{16} = 4$ bits

HW: Calculate I for the above messages in nats and Hartley

Digital Communication System:



Entropy and rate of Information of an Information Source / Model of a Mark off Source

1.3 Average Information Content of Symbols in Long Independence Sequences

Suppose that a source is emitting one of M possible symbols s_0, s_1 ….. s_M in a statically independent sequence

Let p_1, p_2, …….. p_M be the problems of occurrence of the M-symbols resply. suppose further that during a long period of transmission a sequence of N symbols have been generated.

On an average – s_1 will occur NP_1 times

S_2 will occur NP_2 times

: :

s_i will occur NP_i times

The information content of the i_{th} symbol is $I(s_i) = \log \frac{1}{p_i}$ bits

$\therefore$ P_iN occurrences of s_i contributes an information content of

$$i \quad i\,i \quad \log \frac{1}{p_i}$$

$\therefore$ Total information content of the message is = Sum of the contribution due to each of $\text{bits} PN.I(s)=PN.\log$

M symbols of the source alphabet

$\sum_{i=1}^{M} \frac{1}{p_i}$ bits

$\therefore$ Averageinforamtion content $H = \frac{I_{total}}{N} = \sum_{i=1}^{M} \frac{1}{p_i}$ bits per symbol ---- IV

per symbol in given by

This is equation used by Shannon

Average information content per symbol is also called the source entropy.

1.4 The average information associated with an extremely unlikely message, with an extremely likely message and the dependence of H on the probabilities of messages

consider the situation where you have just two messages of probs. 'p' and '(1-p)'.

Average information per message is $H = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$

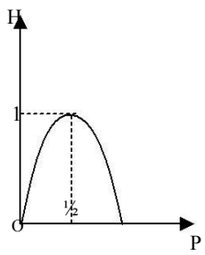
At p = O, H = O and at p = 1, H = O again,

The maximum value of 'H' can be easily obtained as,

$H_{max} = ½ \log_2 2 + ½ \log_2 2 = \log_2 2 = 1$

$\therefore H_{max} = 1$ bit / message

Plot and H can be shown below



The above observation can be generalized for a source with an alphabet of M symbols.

Entropy will attain its maximum value, when the symbol probabilities are equal,

i.e., when $p_1 = p_2 = p_3 = \ldots\ldots\ldots\ldots = p_M = \frac{1}{M}$

$\therefore$ $H_{max} = \log_2 M$ bits / symbol

$H_{max} = \sum p_M \log \frac{1}{p_M}$

$H_{max} = \sum p_M \log \frac{1}{\frac{1}{M}}$

$\therefore H_{max} = \sum_{M}^{1} \log_2 M = \log_2 M$

- **Information rate**

If the source is emitting symbols at a fixed rate of ''r_s' symbols / sec, the average source information rate 'R' is defined as –

$R = r_s . H$ bits / sec

- **Illustrative Examples**

1. Consider a discrete memoryless source with a source alphabet A = { s_o, s_1, s_2} with respective probs. p_0 = ¼, p_1 = ¼, p_2 = ½. Find the entropy of the source.

Solution: By definition, the entropy of a source is given by

$$H = \sum_{i=1}^{M} p_i \log \frac{1}{p_i} \text{ bits/ symbol}$$

H for this example is

$$H(A) = \sum_{i=0}^{2} p_i \log \frac{1}{p_i}$$

Substituting the values given, we get

$$H(A) = p_o \log \frac{1}{P_o} + P_1 \log \frac{1}{p_1} + p_2 \log \frac{1}{p_2}$$

$$= ¼ \log_2 4 + ¼ \log_2 4 + ½ \log_2 2$$

$$= \frac{3}{2} = 1.5 \text{ bits}$$

if r_s = 1 per sec, then

$H'(A) = r_s$ H (A) = 1.5 bits/sec

2. An analog signal is band limited to B Hz, sampled at the Nyquist rate, and the samples are quantized into 4-levels. The quantization levels Q_1, Q_2, Q_3, and Q_4 (messages) are assumed independent and occur with probs.

$P_1 = P_2 = \frac{1}{8}$ and $P_2 = P_3 = \frac{3}{8}$. Find the information rate of the source.

Solution: By definition, the average information H is given by

$$H = p_1 \log \frac{1}{p_1} + p_2 \log \frac{1}{p_2} + p_3 \log \frac{1}{p_3} + p_4 \log \frac{1}{p_4}$$

Substituting the values given, we get

$$H = \frac{1}{8} \log 8 + \frac{3}{8} \log \frac{8}{3} + \frac{3}{8} \log \frac{8}{3} + \frac{1}{8} \log 8$$

$$= 1.8 \text{ bits/ message.}$$

Information rate of the source by definition is

$R = r_s \quad H$

$R = 2B, (1.8) = (3.6\ B)$ bits/sec

3. Compute the values of H and R, if in the above example, the quantities levels are so chosen that they are equally likely to occur,

Solution:

Average information per message is

$H = 4\ (¼ \log_2 4) = 2$ bits/message

and $R = r_s\ H = 2B\ (2) = (4B)$ bits/sec

1.5 Mark off Model for Information Sources

Assumption

A source puts out symbols belonging to a finite alphabet according to certain probabilities depending on preceding symbols as well as the particular symbol in question.

- **Define a random process**

A statistical model of a system that produces a sequence of symbols stated above is and which is governed by a set of probs. is known as a random process.

Therefore, we may consider a discrete source as a random process

And the converse is also true.

i.e. A random process that produces a discrete sequence of symbols chosen from a finite set may be considered as a discrete source.

- **Discrete stationary Mark off process?**

Provides a statistical model for the symbol sequences emitted by a discrete source.

General description of the model can be given as below:

1. At the beginning of each symbol interval, the source will be in the one of 'n' possible states 1, 2, ….. n

   Where 'n' is defined as

$n \leq (M)^m$

M = no of symbol / letters in the alphabet of a discrete stationery source,

m = source is emitting a symbol sequence with a residual influence lasting

'm' symbols.

i.e. m: represents the order of the source.

m = 2 means a 2^{nd} order source

m = 1 means a first order source.

The source changes state once during each symbol interval from say i to j. The probabilityy of this transition is P_{ij}. P_{ij} depends only on the initial state i and the final state j but does not depend on the states during any of the preceeding symbol intervals.

2. When the source changes state from i to j it emits a symbol.

   Symbol emitted depends on the initial state i and the transition ij.

3. Let $s_1, s_2, ... s_M$ be the symbols of the alphabet, and let $x_1, x_2, x_3, ... x_k, ...$ be a sequence of random variables, where x_k represents the k^{th} symbol in a sequence emitted by the source.

   Then, the probability that the k^{th} symbol emitted is s_q will depend on the previous symbols $x_1, x_2, x_3, ..., x_{k-1}$ emitted by the source.

   i.e., $P(X_k = s_q / x_1, x_2, ..., x_{k-1})$

4. The residual influence of

   $x_1, x_2, ..., x_{k-1}$ on x_k is represented by the state of the system at the beginning of the k^{th} symbol interval.

   i.e. $P(x_k = s_q / x_1, x_2, ..., x_{k-1}) = P(x_k = s_q / S_k)$

When S_k in a discrete random variable representing the state of the system at the beginning of the k^{th} interval.

Term 'states' is used to remember past history or residual influence in the same context as the use of state variables in system theory / states in sequential logic circuits.

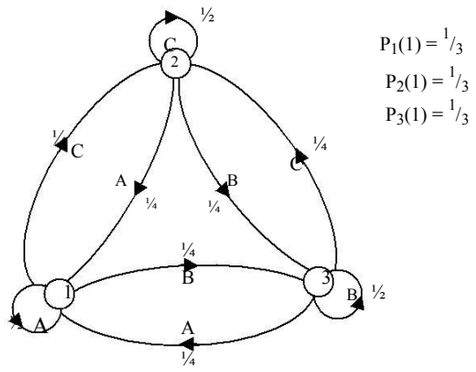
System Analysis with regard to Markoff sources

Representation of Discrete Stationary Markoff sources:

- Are represented in a graph form with the nodes in the graph to represent states and the transition between states by a directed line from the initial to the final state.

- Transition probs. and the symbols emitted corresponding to the transition will be shown marked along the lines of the graph.

  A typical example for such a source is given below.



- It is an example of a source emitting one of three symbols A, B, and C
- The probability of occurrence of a symbol depends on the particular symbol in question and the symbol immediately proceeding it.
- Residual or past influence lasts only for a duration of one symbol.

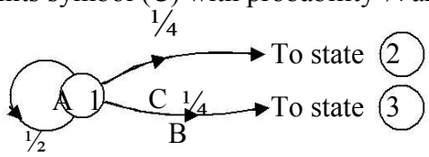
Last symbol emitted by this source

- The last symbol emitted by the source can be A or B or C. Hence past history can be represented by three states- one for each of the three symbols of the alphabet.

- **Nodes of the source**
  - Suppose that the system is in state (1) and the last symbol emitted by the source was A.
  - The source now emits symbol (A) with probability ½and returns to state (1).

    OR
  - The source emits letter (B) with probability ¼ andgoes to state (3)

    OR
  - The source emits symbol (C) with probability ¼ andgoes to state (2).

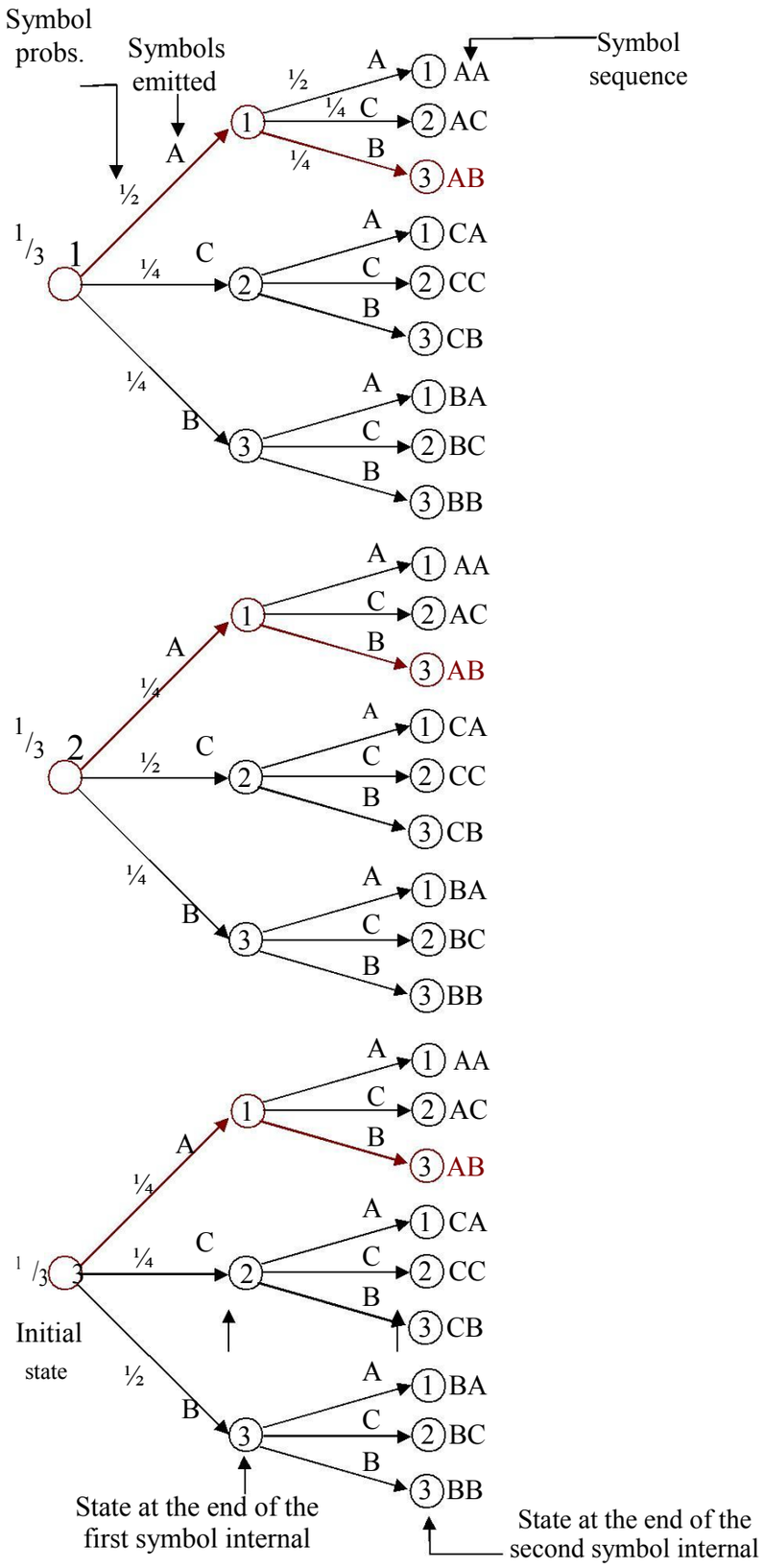


State transition and symbol generation can also be illustrated using a tree diagram.

Tree diagram

- **Tree diagram is a planar graph where the nodes correspond to states and branches correspond to transitions. Transitions between states occur once every T_s seconds.**

  Along the branches of the tree, the transition probabilities and symbols emitted will be indicated.**Tree diagram for the source considered**

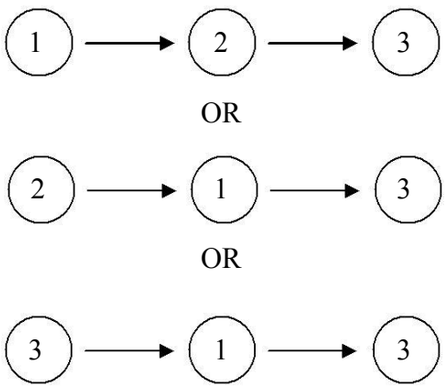

Symbol probs.
Symbols emitted
Symbol sequence
Initial state
State at the end of the first symbol internal
State at the end of the second symbol internal
1/3
1/2
1/4
A
C
B
1
2
3
AA
AC
AB
CA
CC
CB
BA
BC
BB

Use of the tree diagram

Tree diagram can be used to obtain the probabilities of generating various symbol sequences.

Generation a symbol sequence say AB

This can be generated by any one of the following transitions:



Therefore proby of the source emitting the two – s ymbol sequence AB is given by

$$P(AB) = P(S_1 = 1, S_2 = 1, S_3 = 3)$$
$$\text{Or}$$
$$P(S_1 = 2, S_2 = 1, S_3 = 3) \quad \text{----- (1)}$$
$$\text{Or}$$
$$P(S_1 = 3, S_2 = 1, S_3 = 3)$$

Note that the three transition paths are disjoint.

$$\text{Therefore } P(AB) = P(S_1 = 1, S_2 = 1, S_3 = 3) + P(S_1 = 2, S_2 = 1, S_3 = 3) + P(S_1 = 2, S_2 = 1, S_3 = 3) \quad \text{----- (2)}$$

The first term on the RHS of the equation (2) can be written as

$$P(S_1 = 2, S_2 = 1, S_3 = 3)$$
$$= P(S_1 = 1)\, P(S_2 = 1 / S_1 = 1)\, P(S_3 = 3 / S_1 = 1, S_2 = 1)$$
$$= P(S_1 = 1)\, P(S_2 = 1 / S_1 = 1)\, P(S_3 = 3 / S_2 = 1)$$

Recall the Markoff property.

Transition probability to S_3 depends on S_2 but not on how the system got to S_2.

Therefore, $P(S_1 = 1, S_2 = 1, S_3 = 3) = {}^{1}/_{3} \times ½ \times ¼$

Similarly other terms on the RHS of equation (2) can be evaluated.

$$\text{Therefore } P(AB) = {}^{1}\quad {}^{1}\quad {}^{1}/_{3} \times ½ \times ¼ + {}^{1}/_{3} \times ¼ \times ¼ + {}^{1}/_{3} \times ¼ \times ¼ = \frac{4}{48} = \frac{1}{12}$$

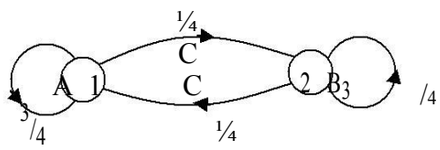
Similarly the probs of occurrence of other symbol sequences can be computed.

Therefore,

In general the probability of the source emitting a particular symbol sequence can be computed by summing the product of probabilities in the tree diagram along all the paths that yield the particular sequences of interest.

Illustrative Example:

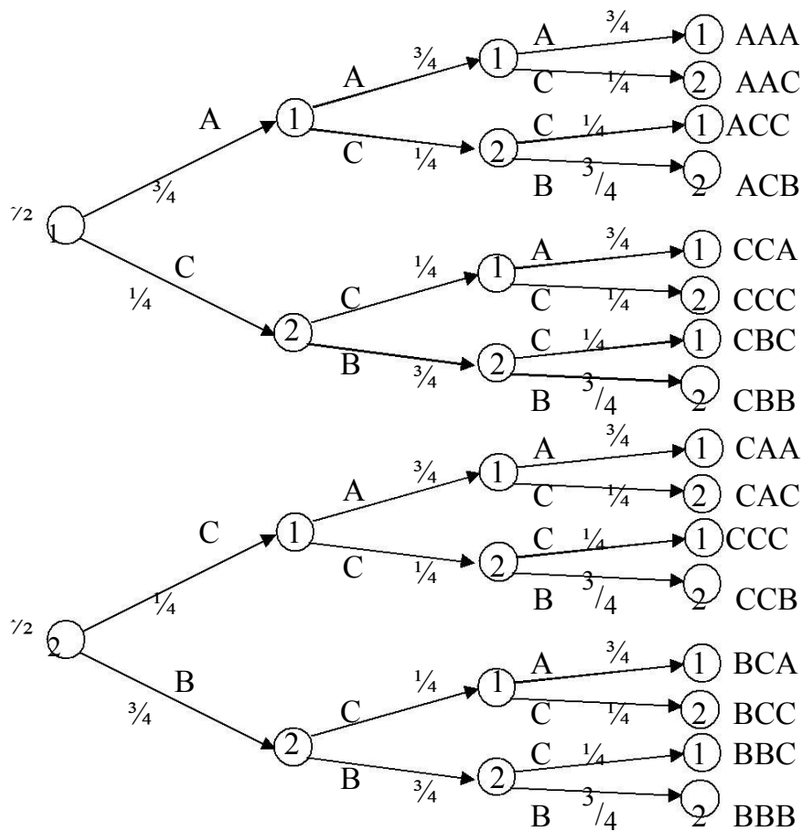
1. For the information source given draw the tree diagram and find the probs. of messages of lengths 1, 2 and 3.



$p_1 = ½$ $P_2 = ½$

Source given emits one of 3 symbols A, B and C

Tree diagram for the source outputs can be easily drawn as shown.



Messages of length (1) and their probs

A ½ x ¾ = $^{3}/_{8}$

B ½ x ¾ = $^{3}/_{8}$

C ½ x ¼ + ½ x ¼ = = ¼ 8 8 $\frac{1+1}{}$

Message of length (2)

How may such messages are there?

Seven

Which are they?

AA, AC, CB, CC, BB, BC & CA

What are their probabilities?

Message AA : ½ x ¾ x ¾ = $\frac{9}{32}$

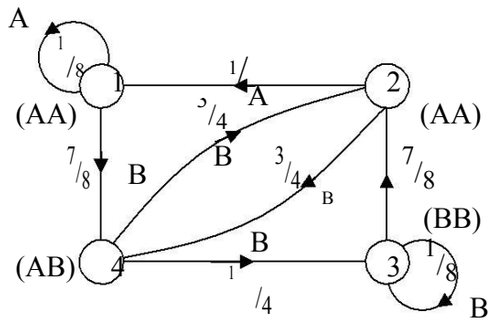
Message AC: ½ x ¾ x ¼ = $\frac{3}{32}$ **and so on.**

Tabulate the various probabilities

Messages of Length (1)	Messages of Length (2)	Messages of Length (3)
A $\frac{3}{8}$	AA $\frac{9}{32}$	AAA $\frac{27}{128}$
B $\frac{3}{8}$	AC $\frac{3}{32}$	AAC $\frac{9}{128}$
C $\frac{1}{4}$	CB $\frac{3}{32}$	ACC $\frac{3}{128}$
	CC $\frac{2}{32}$	ACB $\frac{9}{128}$
	BB $\frac{9}{32}$	BBB $\frac{27}{128}$
	BC $\frac{3}{32}$	BBC $\frac{9}{128}$
	CA $\frac{3}{32}$	BCC $\frac{3}{128}$
		BCA $\frac{9}{128}$
		CCA $\frac{3}{128}$
		CCB $\frac{3}{128}$
		CCC $\frac{2}{128}$
		CBC $\frac{3}{128}$
		CAC $\frac{3}{128}$
		CBB $\frac{9}{128}$
		CAA $\frac{9}{128}$

- **A second order Markoff source**

Model shown is an example of a source where the probability of occurrence of a symbol depends not only on the particular symbol in question, but also on the two symbols proceeding it.



$P_1 \quad (1)\frac{2}{18}$

$P_2 \quad (1)\frac{7}{18}$

$P_3 \quad (1)\frac{7}{18}$

$P_4 \quad (1)\frac{2}{18}$

No. of states: $n \leq (M)^m$;

$4 \leq M^2$

$\therefore M = 2$

m = No. of symbols for which the residual influence lasts
(duration of 2 symbols)
or
M = No. of letters / symbols in the alphabet.

Say the system in the state 3 at the beginning of the symbols emitted by the source were BA.

Similar comment applies for other states.

1.6 Entropy and Information Rate of Markoff Sources

- **Definition of the entropy of the source**

Assume that, the probability of being in state i at he beginning of the first symbol interval is the same as the probability of being in state i at the beginning of the second symbol interval, and so on.

The probability of going from state i to j also doesn't depend on time, Entropy of state 'i' is defined as the average information content of the symbols emitted from the i-th state.

$$H_i = \sum_{j=1}^{n} p_{ij} \log_2 \frac{1}{p_{ij}} \text{ bits / symbol} \qquad \text{------ (1)}$$

Entropy of the source is defined as the average of the entropy of each state.

i.e. $H = E(H_i) = \sum_{j=1}^{n} p_i H_i$ ------ (2)

Where,

Pi = the proby that the source is in state 'i'.

Using eqn (1), eqn. (2) becomes,

$$H = \sum_{i=1}^{n} p_i \sum_{j=1}^{n} p_{ij} \log \frac{1}{p_{ij}} \text{ bits / symbol} \quad \text{------ (3)}$$

Average information rate for the source is defined as

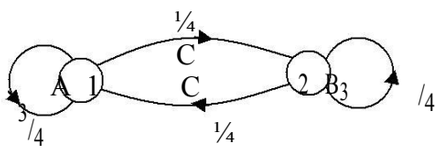
$R = r_s \cdot H$ bits/sec

Where, 'r_s' is the number of state transitions per second or the symbol rate of the source.

The above concepts can be illustrated with an example

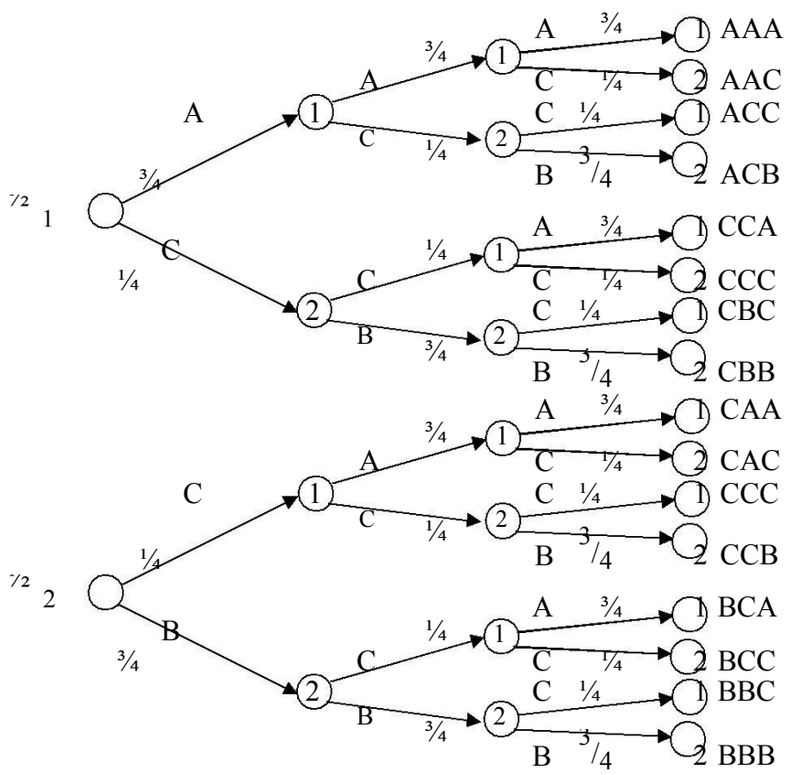
Illustrative Example:

1. Consider an information source modeled by a discrete stationary Mark off random process shown in the figure. Find the source entropy H and the average information content per symbol in messages containing one, two and three symbols.



$p_1 = ½$ $P_2 = ½$

- The source emits one of three symbols A, B and C.
- A tree diagram can be drawn as illustrated in the previous session to understand the various symbol sequences and their probabilities.



As per the outcome of the previous session we have

Messages of Length (1)	Messages of Length (2)	Messages of Length (3)
A 3/8	AA 9/32	AAA 27/128
B 3/8	AC 3/32	AAC 9/128
C 1/4	CB 3/32	ACC 3/128
	CC 2/32	ACB 9/128
	BB 9/32	BBB 27/128
	BC 3/32	BBC 9/128
	CA 3/32	BCC 3/128
		BCA 9/128
		CCA 3/128
		CCB 3/128
		CCC 2/128
		CBC 3/128
		CAC 3/128
		CBB 9/128
		CAA 9/128

By definition H_i is given by

$$H_i = \sum_{j=1}^{n} p_{ij} \log \frac{1}{p_{ij}}$$

Put i = 1,

$$H_i = \sum_{j=1}^{n=2} p_{1j} \log \frac{1}{p_{1j}}$$

$$p_{11} \log \frac{1}{p_{11}} + p_{12} \log \frac{1}{p_{12}}$$

Substituting the values we get,

$$H_1 = \frac{3}{4} \log_2 \frac{1}{(3/4)} + \frac{1}{4} \log_2 \frac{1}{1/4}$$

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

$H_1 = 0.8113$

Similarly $H_2 = \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3} = 0.8113$

By definition, the source entropy is given by,

$$H = \sum_{i=1}^{n} p_i H_i = \sum_{i=1}^{2} p_i H_i$$

$$= \frac{1}{2} (0.8113) + \frac{1}{2} (0.8113)$$

= (0.8113) bits / symbol

To calculate the average information content per symbol in messages containing two symbols.

- **How many messages of length (2) are present? And what is the information content of these messages?**

There are **seven** such messages and their information content is:

$$I(AA) = I(BB) = \log \frac{1}{(AA)} = \log \frac{1}{(BB)}$$

i.e., $I(AA) = I(BB) = \log \frac{1}{(9/32)} = 1.83$ bits

Similarly calculate for other messages and verify that they are

$$I(BB) = I(AC) = \log \frac{1}{} = 3.415$$

$$I(CB) = I(CA) = \underbrace{\qquad\qquad}_{(3/32)\ \text{bits}}$$

$$I(CC) = \log = \frac{1}{P(2/32)} = 4 \text{ bits}$$

- **Computation of the average information content of these messages.**

Thus, we have

$$H_{(two)} = \sum_{i=1}^{7} P_i \log \frac{1}{P_i} \text{ bits / sym.}$$

$$= \sum_{i=1}^{7} P_i \cdot I_i$$

Where I_i = the I's calculated above for different messages of length two

Substituting the values we get,

$$H_{(two)} = \frac{9}{32}(1.83) + \frac{3}{32} \times (3.415) + \frac{3}{32}(3.415) + \frac{2}{32}(4) + \frac{3}{32} \times (3.415) + \frac{3}{32} \times (3.415) + \frac{9}{32} \times (1.83)$$

$$\therefore H_{(two)} = 2.56 \text{ bits}$$

- **Computation of the average information content per symbol in messages containing two symbols using the relation.**

$$G_N = \frac{\text{Average information content of the messages of length N}}{\text{Number of symbols in the message}}$$

Here, N = 2

$$\therefore G_N = \frac{\text{Average information content of the messages of length (2)}}{2}$$

$$= \frac{H_{(two)}}{2}$$

$$= \frac{2.56}{2} = 1.28 \text{ bits / symbol}$$

$$\therefore G_2 = 1.28$$

Similarly compute other G's of interest for the problem under discussion viz G_1 & G_3. You get them as

$G_1 = 1.5612$ bits / symbol

And $G_3 = 1.0970$ bits / symbol

- **from the values of G's calculated**

We note that,

$$G_1 > G_2 > G_3 > H$$

- **Statement**

  It can be stated that the average information per symbol in the message reduces as the length of the message increases.

- **The generalized form of the above statement**

  If P (m_i) is the probability of a sequence m_i of **'N'** symbols form the source with the average information content per symbol in the messages of **N** symbols defined by

$$\mathbf{G_N} = \frac{-\sum_i P(m_i)\log P(m_i)}{N}$$

  Where the sum is over all sequences m_i containing **N** symbols, then $\mathbf{G_N}$ is a monotonic decreasing function of **N** and in the limiting case it becomes.

$$\mathbf{\lim_{N \to \infty} G_N = H \text{ bits / symbol}}$$

  Recall H = entropy of the source

  The above example illustrates the basic concept that the average information content per symbol from a source emitting dependent sequence decreases as the message length increases.

- **It can also be stated as,**

  Alternatively, it tells us that the average number of bits per symbol needed to represent a message **decreases** as the message length increases.

  Problems:

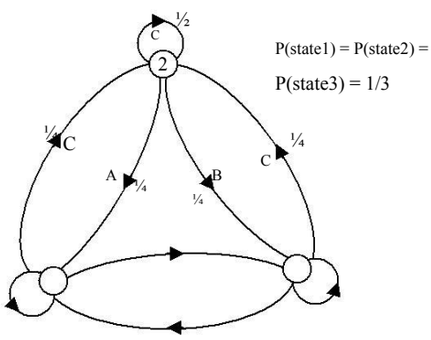
  Example 1

  The state diagram of the stationary Mark off source is shown below

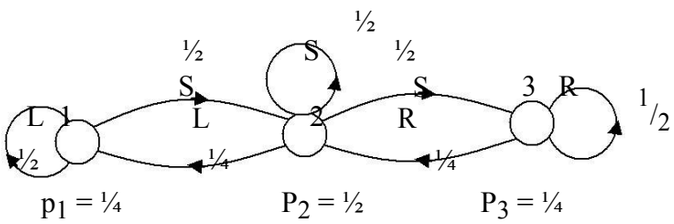
  Find (i) the entropy of each state

  (ii) The entropy of the source

  (iii) G_1, G_2 and verify that $G_1 \geq G_2 \geq H$ the entropy of the source.



For the Mark off source shown, calculate the information rate.



Solution:

By definition, the average information rate for the source is given by

R = r_s . H bits/sec ------ (1)

Where, r_s is the symbol rate of the source

And H is the entropy of the source.

To compute H

Calculate the entropy of each state using,

$$H_i = \sum_{j=1}^{n} p_{ij} \log \frac{1}{p_{ij}} \text{ bits / sym} \quad \text{----- (2)}$$

For this example,

$$H_i = \sum_{j=1}^{3} p_{ij} \log \frac{1}{p_{ij}}; \quad i = 1, 2, 3 \quad \text{------ (3)}$$

Put i = 1

$$\therefore H_i = \sum_{j=1}^{3} p_{1j} \log p_{1j}$$

$= - p_{11} \log p_{11} - p_{12} \log p_{12} - p_{13} \log p_{13}$

Substituting the values, we get

$$H_1 = -\frac{1}{2} \times \log\frac{1}{2} - \frac{1}{2} \log\frac{1}{2} - 0$$

$$= +\left[\frac{1}{2} \log (2) + \frac{1}{2} \log (2)\right]$$

$\therefore$ **H_1 = 1 bit / symbol**

Put i = 2, in eqn. (2) we get,

$$H_2 = - \sum_{j=1}^{3} p_{2j} \log p_{2j}$$

i.e., $H2 = - [p_{21} \log p_{21} + p_{22} \log p_{22} + p_{23} \log p_{23}]$

Substituting the values given we get,

$$H_2 = -\frac{1}{4}\log\frac{1}{4} + \frac{1}{2}\log\frac{1}{2} + \frac{1}{4}\log\frac{1}{4}$$

$$= +\frac{1}{4}\log 4 + \frac{1}{2}\log 2 + \frac{1}{4}\log 4$$

$$= \frac{1}{2}\log 2 + \frac{1}{2} + \log 4$$

$\therefore$ **H_2 = 1.5 bits/symbol**

Similarly calculate H_3 and it will be

H_3 = 1 bit / symbol

With H_i computed you can now compute H, the source entropy, using.

$$H = \sum_{i=1}^{3} P_i \; H_i$$

$$= p_1 H_1 + p_2 H_2 + p_3 H_3$$

Substituting the values we get,

$$H = \frac{1}{4} \times 1 + \frac{1}{2} \times 1.5 + \frac{1}{4} \times 1$$

$$= \frac{1}{4} + \frac{1.5}{2} + \frac{1}{4}$$

$$= \frac{1}{2} + \frac{1.5}{2} = \frac{2.5}{2} = 1.25 \text{ bits / symbol}$$

$\therefore$ H = 1.25 bits/symbol

Now, using equation (1) we have

Source information rate = R = r_s 1.25

Taking 'r_s' as one per second we get

R = 1 x 1.25 = 1.25 bits / sec

Review questions:

(1) Explain the terms (i) Self information (ii) Average information (iii) Mutual Information.

(2) Discuss the reason for using logarithmic measure for measuring the amount of information.

(3) Explain the concept of amount of information associated with message. Also explain what infinite information is and zero information.

(4) A binary source emitting an independent sequence of 0's and 1's with pro babilities p and (1-p) respectively. Plot the entropy of the source.

(5) Explain the concept of information, average information, information rate and redundancy as referred to information transmission.

(6) Let X represents the outcome of a single roll of a fair dice. What is the entropy of X?

(7) A code is composed of dots and dashes. Assume that the dash is 3 times as long as the dot and has one-third the probability of occurrence. (i) Calculate the information in dot and that in a dash; (ii) Calculate the average information in dot-dash code; and (iii) Assume that a dot lasts for 10 ms and this same time interval is allowed between symbols. Calculate the average rate of information transmission.

(8) What do you understand by the term extension of a discrete memory less source? Show that the entropy of the nth extension of a DMS is n times the entropy of the original source.

(9) A card is drawn from a deck of playing cards. A) You are informed that the card you draw is spade. How much information did you receive in bits? B) How much information did you receive if you are told that the card you drew is an ace? C) How much information did you receive if you are told that the card you drew is an ace of spades? Is the information content of the message "ace of spades" the sum of the information contents of the messages "spade" and "ace"?

(10) A block and white TV picture consists of 525 lines of picture information. Assume that each consists of 525 picture elements and that each element can have 256 brightness levels. Pictures are repeated the rate of 30/sec. Calculate the average rate of information conveyed by a TV set to a viewer.

(11) A zero memory source has a source alphabet S= {S1, S2, S3} with P= {1/2, 1/4, 1/4}. Find the entropy of the source. Also determine the entropy of its second extension and verify that H (S^2) = 2H(S).

(12) Show that the entropy is maximum when source transmits symbols with equal probability. Plot the entropy of this source versus p (0<p<1).

(13) The output of an information source consists OF 128 symbols, 16 of which occurs with probability of 1/32 and remaining 112 occur with a probability of 1/224. The source emits 1000 symbols/sec. assuming that the symbols are chosen independently; find the rate of information of the source.

3 BB

A 1 AA
C

Unit - 2: SOURCE CODING

Syllabus:

Encoding of the source output, Shannon's encoding algorithm. Communication Channels, Discrete communication channels, Continuous channels.

Text Books:

- Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996.

Reference Books:

- Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

Unit - 2: SOURCE CODING

2.1 Encoding of the Source Output:

- **Need for encoding**

Suppose that, M – messages = 2^N, which are equally likely to occur. Then recall that average information per messages interval in H = N.

Say further that each message is coded into **N** bits,

$\therefore$ Average information carried by an individual bit is = $\frac{H}{N}$ = 1 bit

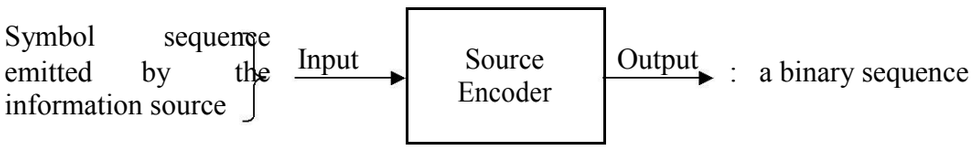
If the messages are not equally likely, then 'H' will be les s than 'N' and each bit will carry less than one bit of information.

- **Is it possible to improve the situation?**

Yes, by using a code in which not all messages are encoded into the same number of bits. The more likely a message is, the fewer the number of bits that should be used in its code word.

- **Source encoding**

Process by which the output of an information source is converted into a binary sequence.



Where, $G_N = -$ $\frac{1}{N}\sum_i p(m_i)\log p(m_i)$

- **If the encoder operates on blocks of 'N' symbols, t he bit rate of the encoder is given as**

Produces an average bit rate of G_N bits / symbol

$p(m_i)$ = Probability of sequence **'m_i'** of **'N'** symbols from the source,

Sum is over all sequences 'm_i' containing **'N'** symbols.

G_N in a monotonic decreasing function of **N** and

$\lim_{N \to \infty} G_N = H$ bits / symbol

Performance measuring factor for the encoder

Coding efficiency: η_c

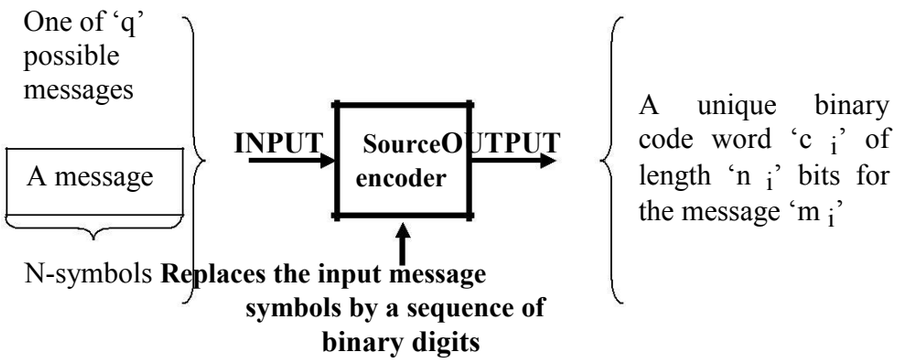
Definition of $\eta c = \frac{\text{Source inf ormation rate}}{\text{Average output bit rate of the encoder}}$

$$\eta_c = \frac{H(S)}{\hat{H}_N}$$

2.2 Shannon's Encoding Algorithm:

- **Formulation of the design of the source encoder**

Can be formulated as follows:



'q' messages : $m_1, m_2, ...m_i, ..., m_q$

Probs. of messages : $p_1, p_2, ...p_i, ..., p_q$

n_i : an integer

- **The objective of the designer**

To find 'n_i' and 'c_i' for i = 1, 2, ..., q such that the average number of bits per symbol $\hat{H}_N$ used in the coding scheme is as close to G_N as possible.

Where, $\hat{H}_N = \frac{1}{N}\sum_{i=1}^{q} n_i p_i$

and $G_N = \frac{1}{N}\sum_{i=1}^{q} p_i \log \frac{1}{p_i}$

i.e., the objective is to have

$\hat{H}_N \to G_N$ as closely as possible

- **The algorithm proposed by Shannon and Fano**

Step 1: Messages for a given block size (N) $m_1, m_2, ... m_q$ are to be arranged in decreasing order of probability.

Step 2: The number of 'n_i' (an integer) assigned to message m_i is bounded by

$$\log_2 \frac{1}{p_i} < n_i < 1 + \log_2 \frac{1}{p_i}$$

Step 3: The code word is generated from the binary fraction expansion of **'F_i'** defined as

$\mathbf{F_i} = \sum_{k=1}^{i-1} p_k$, with $\mathbf{F_1}$ taken to be zero.

Step 4: Choose **'n $_i$'** bits in the expansion of step – (3)

Say, i = 2, then if n_i as per step (2) is = 3 and

If the $\mathbf{F_i}$ as per stop (3) is 0.0011011

Then step (4) says that the code word is: 001 for message (2)

With similar comments for other messages of the source.

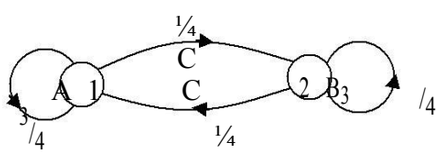
The codeword for the message **'m_i'** is the binary fraction expansion of $\mathbf{F_i}$ upto ' **n_i'** bits.

i.e., $\mathbf{C_i} = \mathbf{(F_i)_{binary,\ ni\ bits}}$

Step 5: Design of the encoder can be completed by repeating the above steps for all the messages of block length chosen.

- **Illustrative Example**

Design of source encoder for the information source given,



$p_1 = ½$ $P_2 = ½$

Compare the average output bit rate and efficiency of the coder for N = 1, 2 & 3.

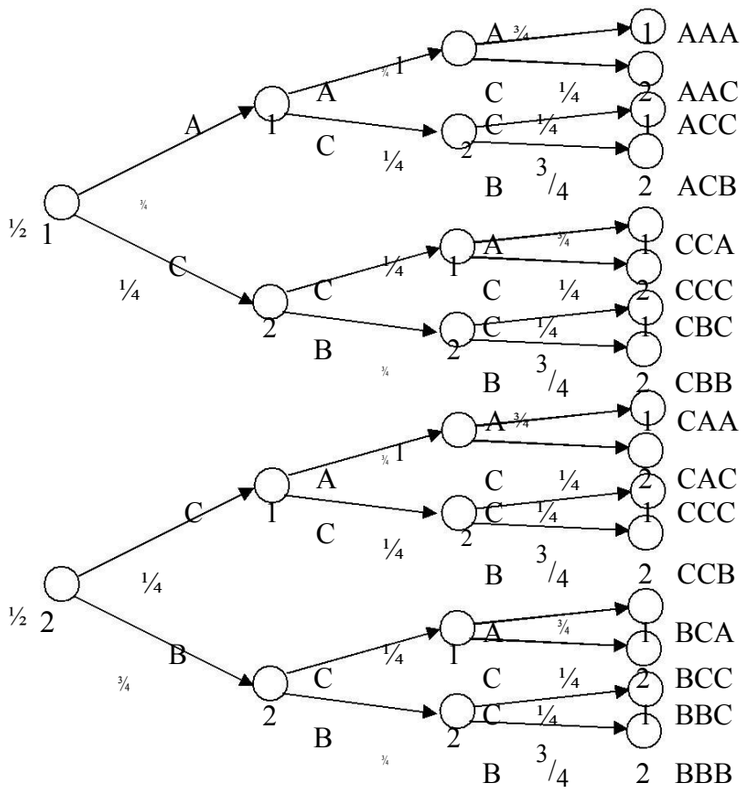
Solution:

The value of 'N' is to be specified.

Case – I: Say N = 3 ← Block size

Step 1: Write the tree diagram and get the symbol sequence of length = 3.

Tree diagram for illustrative example – (1) of session (3)



From the previous session we know that the source emits **fifteen (15)** distinct three symbol messages. They are listed below:

Messages	AAA AAC ACC ACB BBB BBC BCC BCA CCA CCB CCC CBC CAC CBB CAA
Probability	27/128 9/128 3/128 9/128 27/128 9/128 3/128 9/128 3/128 3/128 2/128 3/128 3/128 9/128 9/128

Step 2: Arrange the messages 'm_i' in decreasing order of probability.

Messages m_i	AAA BBB CAA CBB BCA BBC AAC ACB CBC CAC CCB CCA BCC ACC CCC
Probability p_i	27/128 27/128 9/128 9/128 9/128 9/128 9/128 9/128 3/128 3/128 3/128 3/128 3/128 3/128 2/128

Step 3: Compute the number of bits to be assigned to a message 'm_i' using.

$$\log_2 \frac{1}{p_i} < n_i < 1 + \log_2 \frac{1}{p_i}; \; i = 1, 2, \ldots\ldots 15$$

Say i = 1, then bound on 'n_i' is

$$\log_2 \frac{128}{27} < n_1 < 1 + \log_2 \frac{128}{27}$$

i.e., $2.245 < n_1 < 3.245$

Recall 'n_i' has to be an integer

$\therefore$ n_1 can be taken

as, $n_1 = 3$

Step 4: Generate the codeword using the binary fraction expansion of F_i defined as

$F_i = \sum_{k=1}^{i-1} p_k$; with $F_1 = 0$

Say i = 2, i.e., the second message, then calculate n_2 you should get it as 3 bits.

Next, calculate $F_2 = \sum_{k=1}^{2-1} p_k = \sum^{1} p_k = \frac{27}{128}$. Get the binary fraction expansion of $\frac{27}{128}$. You get it as : 0.0011011

Step 5: Since $n_i = 3$, truncate this exploration to 3 – bits.

$\therefore$ **The codeword is: 001**

Step 6: Repeat the above steps and complete the design of the encoder for other messages listed above.

The following table may be constructed

Message m_i	**p_i**	**F_i**	**n_i**	**Binary expansion of F_i**	**Code word c_i**
AAA	$\frac{27}{128}$	0	3	.00000	000
BBB	$\frac{27}{128}$	27/128	3	.001101	001
CAA	$\frac{9}{128}$	54/128	4	0110110	0110
CBB	$\frac{9}{128}$	63/128	4	0111111	0111
BCA	$\frac{9}{128}$	72/128	4	.1001100	1001
BBC	$\frac{9}{128}$	81/128	4	1010001	1010
AAC	$\frac{9}{128}$	90/128	4	1011010	1011
ACB	$\frac{3}{128}$	99/128	4	1100011	1100

CBC	$\frac{3}{128}$	108/128	6	110110	110110
CAC	$\frac{3}{128}$	111/128	6	1101111	110111
CCB	$\frac{3}{128}$	114/128	6	1110010	111001
CCA	$\frac{3}{128}$	117/128	6	1110101	111010
BCC	$\frac{2}{128}$	120/128	6	1111000	111100
ACC		123/128	6	1111011	111101
CCC		126/128	6	1111110	111111

- **the average number of bits per symbol used by the encoder**

  Average number of bits = $\sum n_i \; p_i$

  Substituting the values from the table we get, Average Number of bits = 3.89

  $\therefore$ Average Number of bits per symbol = $\hat{H}_N = \frac{\sum_i n_i}{N}$

  Here N = 3,

  $\therefore \; \hat{H}_3 = \frac{3.89}{3} = 1.3$ bits / symbol

  State entropy is given by

  $$H_i = \sum_{j=1}^{n} p_{ij} \log \frac{1}{p_{ij}} \text{ bits / symbol}$$

  Here number of states the source can be in are two i.e., n = 2

  $$H_i = \sum_{j=1}^{2} p_{ij} \log \frac{1}{p_{ij}}$$

  Say i = 1, then entropy of state – (1) is

  $$H_i = \sum_{j=1}^{2} p_{ij} \log \frac{1}{p_{1j}} = p_{11} \log \frac{1}{p_{11}} + p_{12} \log \frac{1}{p_{12}}$$

  Substituting the values known we get,

$$H_1 = \frac{3}{4} \times \log\left(\frac{1}{3/4}\right) + \frac{1}{4} \log \frac{1}{1/4}$$

$$= \frac{3}{4} \times \log\left(\frac{4}{3}\right) + \frac{1}{4} \log (4)$$

$\therefore$ **$H_1 = 0.8113$**

Similarly we can compute, H_2 as

$$H_2 = \sum_{j=1}^{2} p_{21} \log \frac{1}{p_{21}} = p_{21} + p_{22} \log \frac{1}{p_{22}}$$

Substituting we get,

$$H_2 = \frac{1}{4} \times \log \frac{1}{(1/4)} + \frac{3}{4} \log \frac{1}{3/4}$$

$$= \frac{1}{4} x \log (4) + \frac{3}{4} \log \frac{4}{3}$$

$H_2 = 0.8113$

Entropy of the source by definition is

$$H = \sum_{j=1}^{n} p_i \; H_i \; ;$$

P_i = Probability that the source is in the i^{th} state.

$$H = \sum_{i=1}^{2} p_i \; H_i \; ; \; = p_1H_1 + p_2H_2$$

Substituting the values, we get,

$H = ½ \times 0.8113 + ½ \times 0.8113 = 0.8113$

$\therefore$ **H = 0.8113 bits / sym.**

- **What is the efficiency of the encoder?**

  By definition we have

$$\eta_c = = \frac{H}{\hat{H}_2} x\ 100 = \frac{H}{\hat{H}_3} x\ 100 = \frac{0.8113}{1.3} x\ 100 = 62.4\%$$

$\therefore \eta_c$ for N = 3 is, 62.4%

Case – II

Say N = 2

The number of messages of length 'two' and their probabilities (obtai ned from the tree diagram) can be listed as shown in the table.

Given below

N = 2			
Message	**p_i**	**n_i**	**c_i**
AA	9/32	2	00
BB	9/32	2	01
AC	3/32	4	1001
CB	3/32	4	1010
BC	3/32	4	1100
CA	3/32	4	1101
CC	2/32	4	1111

Calculate $\hat{H}_N$ and verify that it is 1.44 bits / sym.

$\therefore$ Encoder efficiency for this case is

$$\eta_c = \frac{H}{\hat{H}_N} x100$$

Substituting the values we get,

η_c = 56.34%

Case – III: N = 1

Proceeding on the same lines you would see that

N = 1			
Message	**p_i**	**n_i**	**c_i**
A	3/8	2	00
B	3/8	2	01
C	1/4	2	11

$\hat{H}_1$ = 2 bits / symbol and

η_c = 40.56%

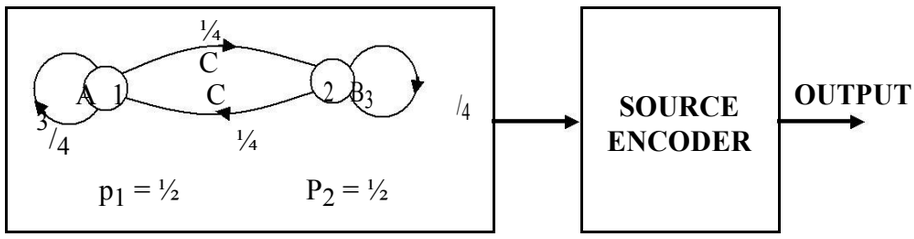
- **Conclusion for the above example**

We note that the average output bit rate of the encoder $\hat{H}_N$ **decreases** as **'N'** increases and

hence the efficiency of the encoder increases as **'N'** increases.

Operation of the Source Encoder Designed:

I. Consider a symbol string ACBBCAAACBBB at the encoder input. If the encoder uses a block size of 3, find the output of the encoder.



INFORMN. SOURCE

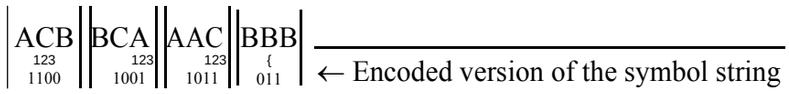
Recall from the outcome of session (5) that for the source given possible three symbol sequences and their corresponding code words are given by –

Message m_i	**n_i**	**Codeword c_i**
AAA	3	000
BBB	3	**001**
CAA	4	0110
CBB	4	**0111**
BCA	4	**1001**
BBC	4	1010
AAC		
ACB	4	**1100**
CBC	6	110110
CAC	6	110111
CCB	6	111001
CCA	6	111010
BCC	6	111100
ACC	6	111101

Determination of the code words and their size as illustrated in the previous session

CCC	6	111111

Output of the encoder can be obtained by replacing successive groups of three **input** symbols by the code words shown in the table. Input symbol string is

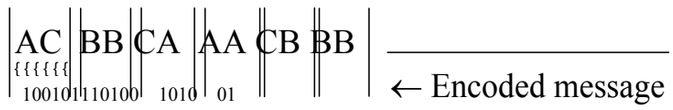


II. If the encoder operates on two symbols at a time what is the output of the encoder for the same symbol string?

Again recall from the previous session that for the source given, different two-symbol sequences and their encoded bits are given by

N = 2		
Message m_i	**No. of bits** n_i	c_i
AA	2	00
BB	2	01
AC	4	1001
CB	4	1010
BC	4	1100
CA	4	1101
CC	4	1111

For this case, the symbol string will be encoded as –



DECODING

- **How is decoding accomplished?**

By starting at the left-most bit and making groups of bits with the codewords listed in the table.

Case – I: N = 3

i) Take the first 3 – bit group viz 110 why?

ii) Check for a matching word in the table.

iii) If no match is obtained, then try the first 4-bit group 1100 and again check for the matching word.

iv) On matching decode the group.

NOTE: For this example, step (ii) is not satisfied and with step (iii) a match is found and the decoding results in **ACB.**

Repeat this procedure beginning with the fifth bit to decode the remaining symbol groups. Symbol string would be ACB BCA AAC BCA

- **Conclusion from the above example with regard to decoding**

  It is clear that the decoding can be done easily by knowing the codeword lengths apriori if no errors occur in the bit string in the transmission process.

- **The effect of bit errors in transmission**

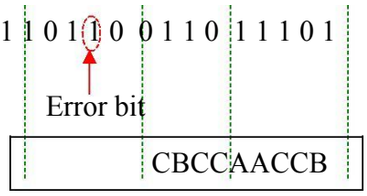
  Leads to serious decoding problems.

Example: For the case of N = 3, if the bit string, 1100100110111001 was received at the decoder input with one bit error as

1101100110111001

What then is the decoded message?

Solution: Received bit string is



----- (1)

For the errorless bit string you have already seen that the decoded symbol string is

ACB BCA AAC BCA **----- (2)**

(1) and (2) reveal the decoding problem with bit error.

Illustrative examples on source encoding

1. **A source emits independent sequences of symbols from a source alphabet containing five symbols with probabilities 0.4, 0.2, 0.2, 0.1 and 0.1.**
   - **i) Compute the entropy of the source**
   - **ii) Design a source encoder with a block size of two.**

Solution: Source alphabet = $(s_1, s_2, s_3, s_4, s_5)$

Probs. of symbols = p_1, p_2, p_3, p_4, p_5

= 0.4, 0.2, 0.2, 0.1, 0.1

(i) Entropy of the source = $H = -\sum_{i=1}^{5} p_i \log p_i$ bits / symbol

Substituting we get,

$H = -[p_1 \log p_1 + p_2 \log p_2 + p_3 \log p_3 + p_4 \log p_4 + p_5 \log p_5]$

$= -[0.4 \log 0.4 + 0.2 \log 0.2 + 0.2 \log 0.2 + 0.1 \log 0.1 + 0.1 \log 0.1]$

H = 2.12 bits/symbol

(ii) Some encoder with N = 2

Different two symbol sequences for the source are:

(s_1s_1)	AA	( ) BB	( ) CC	( ) DD	( ) EE	
(s_1s_2)	AB	( ) BC	( ) CD	( ) DE	( ) ED	
(s_1s_3)	AC	( ) BD	( ) CE	( ) DC	( ) EC	**A total of 25 messages**
(s_1s_4)	AD	( ) BE	( ) CB	( ) DB	( ) EB	
(s_1s_5)	AE	( ) BA	( ) CA	( ) DA	( ) EA	

Arrange the messages in decreasing order of probability and determine the number of bits 'n_i' as explained.

Messages	**Proby. p_i**	**No. of bits n_i**
AA	0.16	3
AB AC BC BA CA	0.08 0.08 0.08 0.08 0.08	4
... ...	0.04 0.04 0.04 0.04 0.04 0.04 0.04	5
... ...	0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02	6
... ...	0.01 0.01 0.01 0.01	7

Calculate $\hat{H}_1$ =

Substituting, $\hat{H}_1$ = 2.36 bits/symbol

2. A technique used in constructing a source encoder consists of arranging the messages in decreasing order of probability and dividing the message into two almost equally probable

groups. The messages in the first group are given the bit 'O' and the messages in the second group are given the bit '1'. The procedure is now applied again for each group separately, and continued until no further division is possible. Using this algorithm, find the code words for six messages occurring with probabilities, 1/24, 1/12, 1/24, 1/6, 1/3, 1/3

Solution: (1) Arrange in decreasing order of probability

m5	1/3	0	0			
m6	1/3	0	1			← 1st division
m4	1/6	1	0			← 2nd division
m2	1/12	1	1	0		← 3rd division
m1	1/24	1	1	1	0	← 4th division
m3	1/24	1	1	1	1	

∴ Code words are

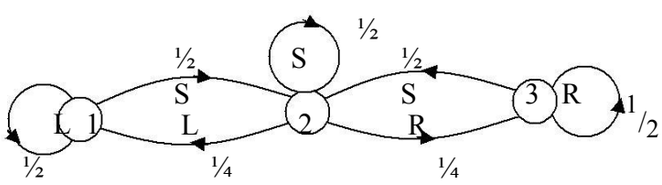
$m_1 = 1110$
$m_2 = 110$
$m_3 = 1111$
$m_4 = 10$
$m_5 = 00$
$m_6 = 01$

Example (3)

a) **For the source shown, design a source encoding scheme using block size of two symbols and variable length code words**

b) **Calculate $\hat{H}_2$ used by the encoder**

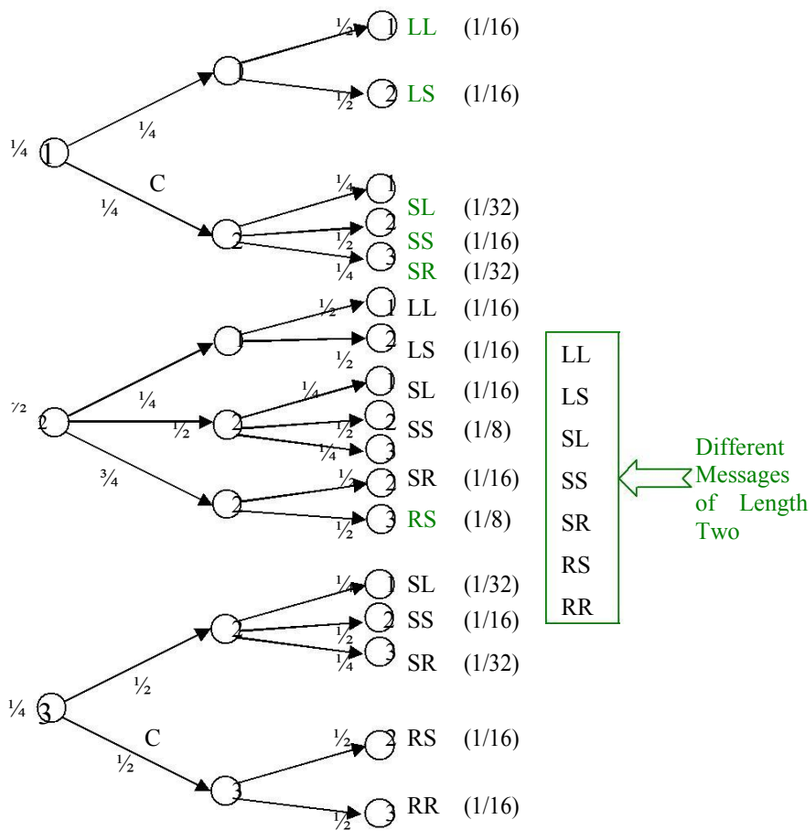
c) **If the source is emitting symbols at a rate of 1000 symbols per second, compute the output bit rate of the encoder.**



$p_1 = ¼$ $P_2 = ½$ $P_3 = ¼$

Solution (a)

1. The tree diagram for the source is



2. Note, there are **seven** messages of length (2). They are SS, LL, LS, SL, SR, RS & RR.
3. Compute the message probabilities and arrange in descending order.
4. Compute n_i, F_i. F_i (in binary) and c_i as explained earlier and tabulate the results, with usual notations.

Message m_i	**p_i**	**n_i**	**F_i**	**F_i (binary)**	**c_i**
SS	1/4	2	0	.0000	00
LL	1/8	3	1/4	.0100	010
LS	1/8	3	3/8	.0110	011
SL	1/8	3	4/8	.1000	100
SR	1/8	3	5/8	.1010	101
RS	1/8	3	6/8	.1100	110
RR	1/8	3	7/8	.1110	111

$G_2 = \frac{1}{2}\sum_{i=1}^{7} -p_i \log_2 p_i = 1.375$ bits/symbol

(b) $\hat{H}_2 = \frac{1}{2}\sum_{i=1}^{7} p_i n_i = 1.375$ bits/symbol

Recall, $\hat{H}_N \le G_N + \frac{1}{N}$; Here N = 2

$\therefore \hat{H}_2 \le G_2 + \frac{1}{2}$

(c) Rate = 1375 bits/sec.

2.3 SOURCE ENCODER DESIGN AND COMMUNICATION CHANNELS

- **The schematic of a practical communication system is shown.**

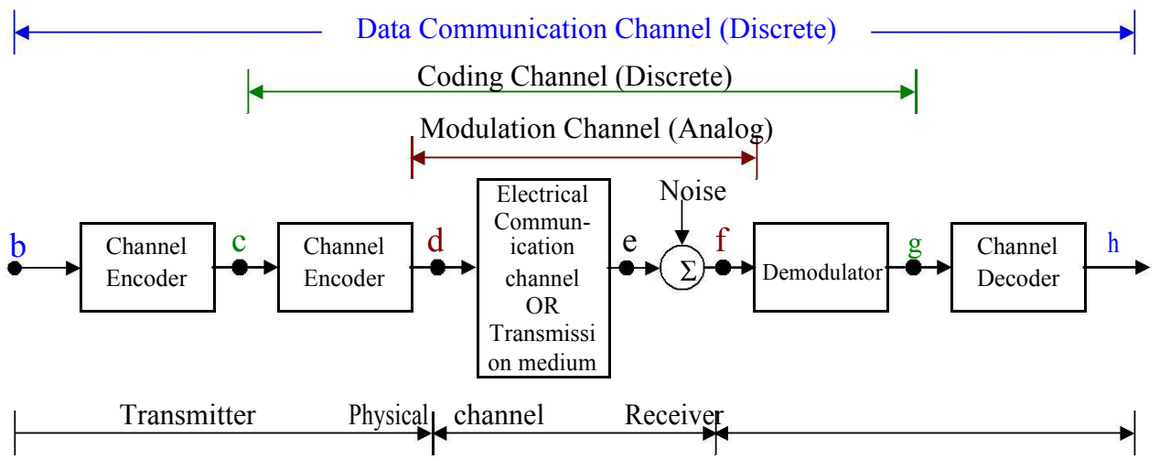


Fig. 1: BINARY COMMN. CHANNEL CHARACTERISATION

- **'Communication Channel'**

Communication Channel carries different meanings and characterizations depending on its terminal points and functionality.

(i) Portion between points c & g:

Referred to as coding channel

Accepts a sequence of symbols at its input and produces a sequence of symbols at its output.

Completely characterized by a set of transition probabilities p_{ij}. These probabilities will depend on the parameters of – (1) The modulator, (2) Transmiss ion media, (3) Noise, and (4) Demodulator

A discrete channel

(ii) Portion between points d and f:

Provides electrical connection between the source and the destination.

The input to and the output of this channel are analog electrical waveforms.

Referred to as 'continuous' or modulation channel or simply analog channel. Are subject to several varieties of impairments –

Due to amplitude and frequency response variations of the channel within the passband.

Due to variation of channel characteristics with time.

Non-linearities in the channel.

Channel can also corrupt the signal statistically due to various types of additive and multiplicative noise.

2.4 Mathematical Model for Discrete Communication Channel:

Channel between points c & g of Fig. – (1)

- **The input to the channel?**

  A symbol belonging to an alphabet of 'M' symbols in the general case is the input to the channel.

- **he output of the channel**

  A symbol belonging to the same alphabet of 'M' input symbols is the output of the channel.

- **Is the output symbol in a symbol interval same as the input symbol during the same symbol interval?**

  The discrete channel is completely modeled by a set of probabilities –

  p_i^t Probability that the input to the channel is the i^{th} symbol of the alphabet.

  (i = 1, 2, ……. M)

  and

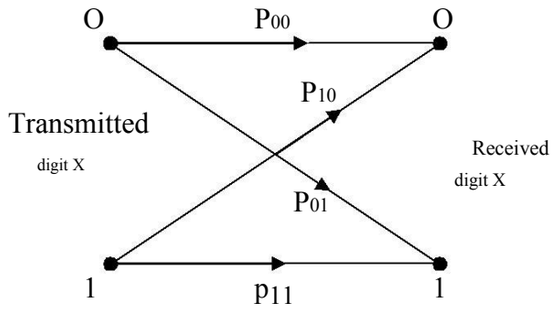
  p_{ij} Probability that the i^{th} symbol is received as the j^{th} symbol of the alphabet at the output of the channel.

- **Discrete M-ary channel**

  If a channel is designed to transmit and receive one of 'M' possibl e symbols, it is called a discrete M-ary channel.

- **discrete binary channel and the statistical model of a binary channel**

Shown in Fig. – (2).



$p_{ij} = p(Y = j / X=i)$

$p_o^t = p(X = o)$; p_1^t $P(X = 1)$

$p_o^r = p(Y = o)$; p_1^r $P(Y = 1)$

$p_{oo} + p_{o1} = 1$; $p_{11} + p_{10} = 1$

Fig. – (2)

- **Its features**

  X & Y: random variables – binary valued

  Input nodes are connected to the output nodes by four paths.

  (i) Path on top of graph : Represents an input 'O' appearing correctly as 'O' as the channel output.

  (ii) Path at bottom of graph :

  (iii) Diogonal path from 0 to 1 : Represents an input bit O appearing incorrectly as 1 at the channel output (due to noise)

  (iv) Diagonal path from 1 to 0 : Similar comments

Errors occur in a random fashion and the occurrence of errors can be statistically modelled by assigning probabilities to the paths shown in figure (2).

- **A memory less channel:**

  If the occurrence of an error during a bit interval does not affect the behaviour of the system during other bit intervals.

  Probability of an error can be evaluated as

  $p(error) = P_e = P(X \neq Y) = P(X = 0, Y = 1) + P(X = 1, Y = 0)$

  $P_e = P(X = 0) . P(Y = 1 / X = 0) + P(X = 1), P(Y = 0 / X= 1)$

  Can also be written as,

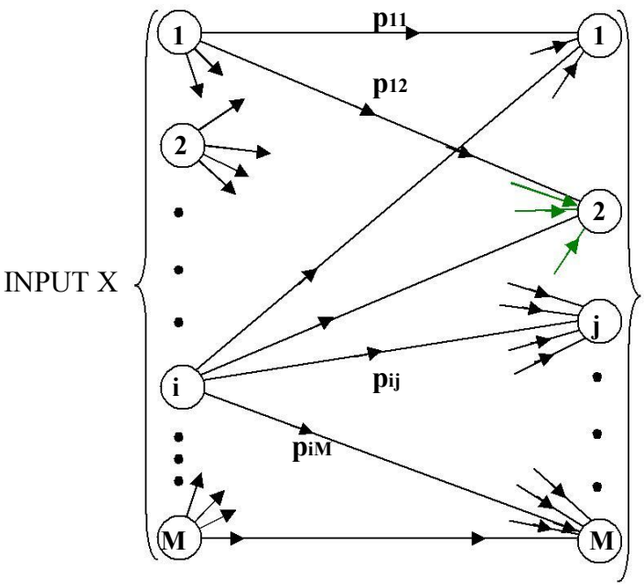
  $$P_e = p_o^t \, p_{01} + p_1^t \, p_{10} \qquad \text{------ (1)}$$

  We also have from the model

$p_o^r = p_o^t p_{00} + p_1^t \cdot p_{10}$, and
$p_1^r = p_o^t p_{01} + p_1^t p_{11}$ ----- (2)

- **Binary symmetric channel (BSC)**
  If, $p_{00} = p_{11} = p$ (say), then the channel is called a BSC.

- **Parameters needed to characterize a BSC**
  - **Model of an M-ary DMC.**



$p_i^t = p(X = i)$
$p_j^r = p(Y = j)$
$p_{ij} = p(Y = j / X = i)$

Fig. – (3)

This can be analysed on the same lines presented above for a binary channel.

$$p_j^r = \sum_{i=1}^{M} p_i^t \, p_{ij} \qquad \text{----- (3)}$$

- **The p(error) for the M-ary channel**
  Generalising equation (1) above, we have

$$P(\text{error}) = P_e = \sum_{i=1}^{M} p_i^t \sum_{\substack{j=1 \\ j \neq i}}^{M} p_{ij} \qquad \text{----- (4)}$$

- **In a DMC how many statistical processes are involved and which are they?**
  Two, (i) Input to the channel and
  (ii) Noise

- **Definition of the different entropies for the DMC.**

i) Entropy of INPUT X: H(X).

$$H(X) = -\sum_{i=1}^{M} p_i^t \log\left(p_i^t\right) \text{ bits / symbol} \quad \text{----- (5)}$$

ii) Entropy of OUTPUT Y: H(Y)

$$H(Y) = -\sum_{i=1}^{M} p_i^r \log\left(p_i^r\right) \text{ bits / symbol} \quad \text{----- (6)}$$

iii) Conditional entropy: H(X/Y)

$$H(X/Y) = -\sum_{i=1}^{M}\sum_{j=1}^{M} P(X=i, Y=j) \log\left(p(X=i/Y=j)\right) \text{ bits/symbol} \quad \text{- (7)}$$

iv) Joint entropy: H(X,Y)

$$H(X, Y) = -\sum_{i=1}^{M}\sum_{i=1}^{M} P(X=i, Y=j) \log\left(p(X=i, Y=j)\right) \text{ bits/symbol} \quad \text{- (8)}$$

v) Conditional entropy: H(Y/X)

$$H(Y/X) = -\sum_{i=1}^{M}\sum_{i=1}^{M} P(X=i, Y=j) \log\left(p(Y=j/X=i)\right) \text{ bits/symbol} \quad \text{- (9)}$$

Representation of the conditional entropy

- H(X/Y) represents how uncertain we are of the channel input ‘x’, on the average, when we know the channel output ‘Y’.

Similar comments apply to H(Y/X)

vi) Joint Entropy H(X, Y) $= H(X) + H(Y/X) = H(Y) + H(X/Y)$ - (10)

ENTROPIES PERTAINING TO DMC

- **To prove the relation for H(X Y)**

By definition, we have,

$$H(XY) = -\sum_{i}^{M}\sum_{j}^{M} p(i, j) \log p(i, j)$$

i associated with variable X, white j with variable Y.

$$H(XY) = -\sum_{ij}\sum p(i)\,p(j/i)\log\left[p(i)\,p(j/i)\right]$$

$$= -\sum_{ij}\sum p(i)\,p(j/i)\log p(i) + \sum\sum p(i)\,p(j/i)\log p(i)$$

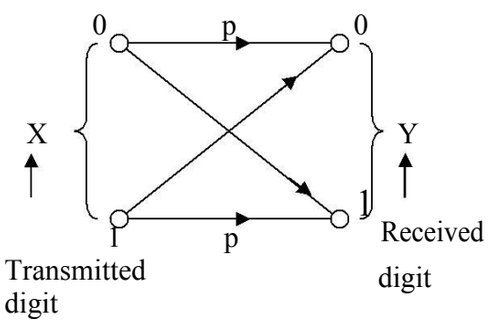
Say, 'i' is held constant in the first summation of the first t erm on RHS, then we can write H(XY) as

$$H(XY) = -\sum p(i)1\log p(i) + \sum\sum p(ij)\log p(j/i)$$

$$\therefore H(XY) = H(X) + H(Y/X)$$

Hence the proof.

1. For the discrete channel model shown, find, the probability of error.



Since the channel is symmetric,
p(1, 0) = p(0, 1) = (1 - p)

Proby. Of error means, situation when X ≠ Y

$$P(\text{error}) = P_e = P(X \neq Y) = P(X = 0, Y = 1) + P(X = 1, Y = 0)$$
$$= P(X = 0)\,.\,P(Y = 1/X = 0) + P(X = 1)\,.\,P(Y = 0/X = 1)$$

Assuming that 0 & 1 are equally likely to occur

$$P(\text{error}) = \frac{1}{2} \times (1 - p) + \frac{1}{2}(1 - p) = \frac{1}{2} - \frac{p}{2} + \frac{1}{2} - \frac{p}{2}$$

$$\therefore \textbf{P(error) = (1 – p)}$$

2. A binary channel has the following noise characteristics:

P(Y/X)		Y	
		0	1
X	0	2/3	1/3
	1	1/3	2/3

If the input symbols are transmitted with probabilities ¾ & ¼ respectively, find H(X), H(Y), H(XY), H(Y/X).

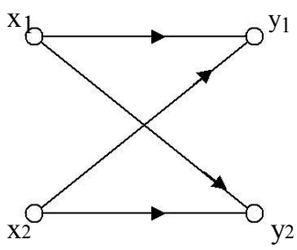
Solution:

Given = P(X = 0) = ¾ and P(Y = 1) ¼

$$\therefore H(X) = -\sum_i p_i \log p_i = \frac{3}{4}\log_2\frac{4}{3} + \frac{1}{4}\log_2 4 = 0.811278 \text{ bits / symbol}$$

Compute the probability of the output symbols.

Channel model is-



$p(Y = Y_1) = p(X = X_1, Y = Y_1) + p(X = X_2, Y = Y_1)$ ----- (1)

To evaluate this construct the p(XY) matrix using.

$$P(XY) = p(X) \cdot p(Y/X) = \begin{matrix} & y_1 & y_2 \\ x_1 & \frac{1}{2} & \frac{1}{4} \\ x_2 & \frac{1}{12} & \frac{1}{6} \end{matrix} \quad \text{----- (2)}$$

$$\therefore P(Y = Y_1) = \frac{1}{2} + \frac{1}{12} = \frac{7}{12}$$ -- Sum of first column of matrix (2)

Similarly $P(Y_2) = \frac{5}{12}$ sum of 2nd column of P(XY)

Construct P(X/Y) matrix using

$P(XY) = p(Y) \cdot p(X/Y)$ i.e., $p(X/Y) = \frac{p(XY)}{p(Y)}$

$$\therefore p\frac{X_1}{Y_1} = \frac{p(X_1Y_1)}{p(Y_1)} = \frac{\frac{1}{2}}{\frac{7}{12}} = \frac{12}{14} = \frac{6}{7} \text{ and so on}$$

$$\therefore p(Y/X) = \begin{matrix} \frac{6}{7} & \frac{3}{5} \\ \frac{1}{7} & \frac{2}{5} \end{matrix} \quad \text{-----(3)}$$

= ?

$$H(Y) = \sum_i p_i \log \frac{1}{p_i} = \frac{7}{12} \log \frac{12}{7} + \frac{5}{12} \log \frac{12}{5} = 0.979868 \text{ bits/sym.}$$

$$H(XY) = \sum_i \sum_j p(XY) \log \frac{1}{p(XY)}$$

$$= \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{12}\log 12 + \frac{1}{6}\log 6$$

$\therefore H(XY) = 1.729573$ bits/sym.

$$H(X/Y) = \sum \sum p(XY) \log \frac{1}{p(X/Y)}$$

$$= \frac{1}{2}\log\frac{7}{6} + \frac{1}{4}\log\frac{5}{3} + \frac{1}{12}\log 7 + \frac{1}{6}\log\frac{5}{2} = 1.562$$

$$H(X/Y) = \sum \sum p(XY) \log \frac{1}{p(Y/X)}$$

$$= \frac{1}{2}\log\frac{3}{2} + \frac{1}{4}\log 3 + \frac{1}{12}\log 3 + \frac{1}{6}\log\frac{3}{2}$$

3. The joint probability matrix for a channel is given below. Compute H(X), H(Y), H(XY), H(X/Y) & H(Y/X)

$$P(XY) = \begin{matrix} 0.05 & 0 & 0.2 & 0.05 \\ 0 & 0.1 & 0.1 & 0 \\ 0 & 0 & 0.2 & 0.1 \\ 0.05 & 0.05 & 0 & 0.1 \end{matrix}$$

Solution:

Row sum of P(XY) gives the row matrix P(X)

$\therefore P(X) = [0.3, 0.2, 0.3, 0.2]$

Columns sum of P(XY) matrix gives the row matrix P(Y)

$\therefore$ P(Y) = [0.1, 0.15, 0.5, 0.25]

Get the conditional probability matrix P(Y/X)

$$P(Y/X) = \begin{matrix} \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{2}{5} & \frac{2}{5} \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{2}{5} \end{matrix}$$

Get the condition probability matrix P(X/Y)

$$\therefore P(X/Y) = \begin{matrix} \frac{1}{2} & 0 & \frac{2}{5} & \frac{1}{5} \\ 0 & \frac{2}{3} & \frac{1}{5} & 0 \\ 0 & 0 & \frac{2}{5} & \frac{2}{5} \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{2}{5} \end{matrix}$$

Now compute the various entropies required using their defining equations.

(i) $H(X) = \sum_i p(X).\log \frac{1}{p(X)} = 2\ 0.3 \log \frac{1}{0.3} + 2\ 0.2 \log \frac{1}{0.2}$

$\therefore$ H (X) = 1.9705 bits / symbol

(ii) $H(Y) = \sum_j p(Y).\log \frac{1}{p(Y)} = 0.1 \log \frac{1}{0.1} + 0.15 \log \frac{1}{0.15}$

$+ 0.5 \log \frac{1}{0.5} + 0.25 \log \frac{1}{0.25}$

$\therefore$ H (Y) = 1.74273 bits / symbol

(iii) $H(XY) = \sum\sum p(XY) \log \frac{1}{p(XY)}$

$= 4\ 0.05 \log \frac{1}{0.05} + 4\ 0.1 \log \frac{1}{0.1} + 2\ 0.2 \log \frac{1}{0.2}$

$\therefore$ **H(XY) = 3.12192**

(iv) $H(X/Y) = \sum\sum p(XY) \log \dfrac{1}{p(X/Y)}$

Substituting the values, we get.

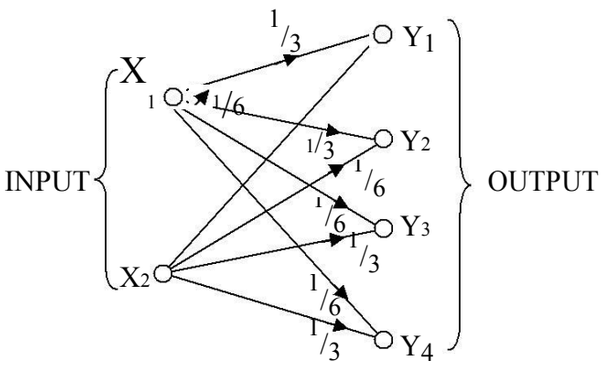
$\therefore$ **H(X/Y) = 4.95 bits / symbol**

(v) $H(Y/X) = \sum\sum p(XY) \log \dfrac{1}{p(Y/X)}$

Substituting the values, we get.

$\therefore$ **H(Y/X) = 1.4001 bits / symbol**

4. Consider the channel represented by the statistical model shown. Write the channel matrix and compute H(Y/X).



For the channel write the conditional probability matrix P(Y/X).

$$P(Y/X) = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} \end{array}$$

NOTE: 2nd row of P(Y/X) is 1st row written in reverse order. If this is the situation, then channel is called a symmetric one.

First row of P(Y/X) . $P(X_1) = \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times \frac{1}{6} + \frac{1}{2} \times \frac{1}{6}$

Second row of P(Y/X) . $P(X_2) = \frac{1}{2} \times \frac{1}{6} + \frac{1}{2} \times \frac{1}{6} + \frac{1}{6} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{3}$

Recall

$P(XY) = p(X), p(Y/X)$

$\therefore P(X_1Y_1) = p(X_1) \cdot p(Y_1X_1) = 1 \cdot \frac{1}{3}$

$P(X_1Y_2) = \frac{1}{3}$, $p(X_1, Y_3) = \frac{1}{6} = (Y_1X_4)$ and so on.

$$P(X/Y) = \begin{matrix} 1 & \frac{1}{6} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{6} & \frac{1}{6} \end{matrix}$$

$$H(Y/X) = \sum\sum p(XY) \log \frac{1}{p(Y/X)}$$

Substituting for various probabilities we get,

$$H(Y/X) = \frac{1}{6}\log 3 + \frac{1}{6}\log 3 + \frac{1}{12}\log 6 + \frac{1}{12}\log 6 + \frac{1}{12}\log 6 + \frac{1}{12}\log 6 + \frac{1}{6}\log 3 + \frac{1}{6}\log 3$$

$$= 4 \times \frac{1}{6}\log 3 + 4 \times \frac{1}{12}\log 6$$

$$= 2 \times \frac{1}{3}\log 3 + \frac{1}{3}\log 6 = \therefore ?$$

5. Given joint proby. matrix for a channel compute the various entropies for the input and output rv's of the channel.

$$P(X \cdot Y) = \begin{matrix} 0.2 & 0 & 0.2 & 0 \\ 0.1 & 0.01 & 0.01 & 0.01 \\ 0 & 0.02 & 0.02 & 0 \\ 0.04 & 0.04 & 0.01 & 0.06 \\ 0 & 0.06 & 0.02 & 0.2 \end{matrix}$$

Solution:

P(X) = row matrix: Sum of each row of P(XY) matrix.

$\therefore$ P(X) = [0.4, 0.13, 0.04, 0.15, 0.28]

P(Y) = column sum = [0.34, 0.13, 0.26, 0.27]

1. $H(XY) = \sum\sum p(XY) \log \frac{1}{p(XY)}$ = 3.1883 bits/sym.

2. $H(X) = \sum p(X) \log \frac{1}{p(X)} = 2.0219$ bits/sym.

3. $H(Y) = \sum p(Y) \log \frac{1}{p(Y)} = 1.9271$ bits/sym.

Construct the p(X/Y) matrix using, p(XY) = p(Y) p(X/Y)

$$\text{or } P(X/Y) = \frac{p(XY)}{p(Y)} = \begin{matrix} \frac{0.2}{0.34} & 0 & \frac{0.2}{0.36} & 0 \\ \frac{0.1}{0.34} & \frac{0.01}{0.13} & \frac{0.01}{0.26} & \frac{0.01}{0.27} \\ 0 & \frac{0.02}{0.13} & \frac{0.02}{0.26} & 0 \\ \frac{0.04}{0.34} & \frac{0.04}{0.13} & \frac{0.01}{0.26} & \frac{0.06}{0.27} \\ 0 & \frac{0.06}{0.13} & \frac{0.02}{0.26} & \frac{0.2}{0.27} \end{matrix}$$

4. $H(X/Y) = -\sum\sum p(XY) \log p(X / Y) = 1.26118$ bits/sym.

Problem:

Construct p(Y/X) matrix and hence compute H(Y/X).

Rate of Information Transmission over a Discrete Channel :

- **For an M-ary DMC, which is accepting symbols at the rate of r_s symbols per second, the average amount of information per symbol going into the channel is given by the entropy of the input random variable 'X'.**

  i.e., $H(X) = -\sum_{i=1}^{M} p_i^t \log_2 p_i^t$ ----- (1)

  Assumption is that the symbol in the sequence at the input to the channel occur in a statistically independent fashion.

- **Average rate at which information is going into the channel is**

  $D_{in} = H(X), r_s$ bits/sec ----- (2)

- **Is it possible to reconstruct the input symbol sequence with certainty by operating on the received sequence?**

- **Given two symbols 0 & 1 that are transmitted at a rate of 1000 symbols or bits per second. With** $p_0^t = \frac{1}{2}$ & $p_1^t = \frac{1}{2}$

  Din at the i/p to the channel = 1000 bits/sec. Assume that the channel is symmetric with the probability of errorless transmission p equal to 0.95.

Rate of transmission of information:

- **Recall H(X/Y) is a measure of how uncertain we are of the input X given output Y.**

- **What do you mean by an ideal errorless channel?**

- **H(X/Y) may be used to represent the amount of information lost in the channel.**

- **Define the average rate of information transmitted over a channel (D_t).**

Amount of information going into the channel − Amount of information lost r_s

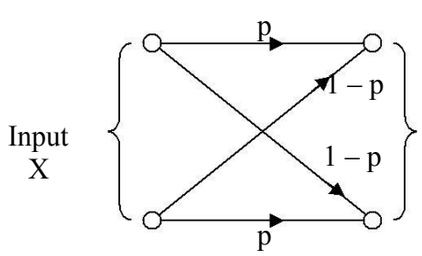
Symbolically it is,

$D_t = [H(H) - H(X/Y)] . r_s$ bits/sec.

When the channel is very noisy so that output is statistically independent of the input, H(X/Y) = H(X) and hence all the information going into the channel is lost and no information is transmitted over the channel.

DISCRETE CHANNELS:

1. **A binary symmetric channel is shown in figure. Find the rate of information transmission over this channel when p = 0.9, 0.8 & 0.6. Assume that the symbol (or bit) rate is 1000/second.**



$p(X = 0) = p(X = 1) = \frac{1}{2}$

Example of a BSC

Solution:

$$\mathbf{H(X)} = \frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2 = 1 \text{ bit / sym.}$$

$$\therefore D_{in}\ r_s\ H(X) = 1000 \text{ bit / sec}$$

By definition we have,

$$\mathbf{D_t = [H(X) - H(X/Y)]}$$

$$\mathbf{Where,\ H(X/Y)} = -\sum_i\sum_j [p(XY)\log p(X/Y)]\ \mathbf{.\ r_s}$$

Where X & Y can take values.

X	Y
0	0
0	1
1	0
1	1

$$\therefore \mathbf{H(X/Y) = - P(X = 0, Y = 0) \log P (X = 0 / Y = 0)}$$

$$\mathbf{= - P(X = 0, Y = 1) \log P (X = 0 / Y = 1)}$$

$$\mathbf{= - P(X = 1, Y = 0) \log P (X = 1 / Y = 0)}$$

$$\mathbf{= - P(X = 1, Y = 1) \log P (X = 1 / Y = 1)}$$

The conditional probability p(X/Y) is to be calculated for all the possible values that X & Y can take.

Say X = 0, Y = 0, then

$$\mathbf{P(X = 0 / Y = 0)} = \frac{p(Y = 0 / X = 0)\ p(X = 0)}{p(Y = 0)}$$

Where

$$\frac{Y = 0}{X = 1}$$

$$= \mathbf{p} \cdot \frac{1}{2} + \frac{1}{2}\ \mathbf{(1 - p)}$$

$\therefore$ **p(Y = 0) =** $\frac{1}{2}$

$\therefore$ **p(X = 0 /Y = 0) = p**

Similarly we can calculate

p(X = 1 / Y = 0) = 1 – p

p(X = 1 / Y = 1) = p

p(X = 0 / Y = 1) = 1 – p

$$\therefore \mathbf{H(X/Y)} = -\left[\frac{1}{2} p \log_2 p + \frac{1}{2}(1-p)\log_2 (1-p) + \frac{1}{2} p \log_2 p + \frac{1}{2}(1-p)\log(1-p)\right]$$

$$= -\left[p \log_2 p + (1-p)\log_2 (1-p)\right]$$

$\therefore$ **D_t rate of inforn. transmission over the channel is = [H(X) – H (X/Y)] . r_s**

with, p = 0.9, D_t = 531 bits/sec.

p = 0.8, D_t = 278 bits/sec.

p = 0.6, D_t = 29 bits/sec.

- **What does the quantity (1 – p) represent?**
- **What do you understand from the above example?**

2. A discrete channel has 4 inputs and 4 outputs. The input probabilities are P, Q, Q, and P. The conditional probabilities between the output and input are.

P(y/x)		Y			
		0	1	2	3
X	0	1	–	–	–
	1	–	p	(1–p)	–
	2	–	(1–p)	(p)	–
	3	–	–	–	1

Write the channel model.

Solution: The channel model can be deduced as shown below:

Given, P(X = 0) = P

P(X = 1) = Q

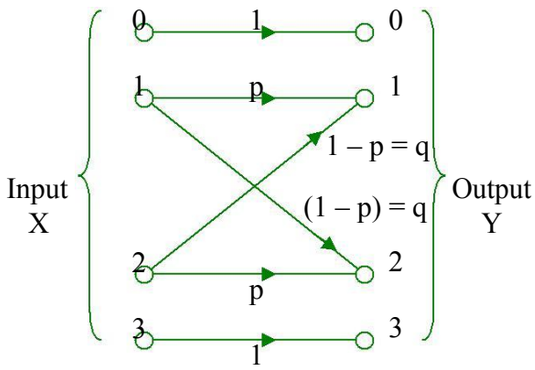
P(X = 2) = Q

P(X = 3) = P

Off course it is true that: P + Q + Q + P = 1

i.e., 2P + 2Q = 1

∴ Channel model is



- **What is H(X) for this?**

  H(X) = - [2P log P + 2Q log Q]

- **What is H(X/Y)?**

  H(X/Y) = - 2Q [p log p + q log q]

  = 2 Q . α

1. **A source delivers the binary digits 0 and 1 with equal probability into a noisy channel at a rate of 1000 digits / second. Owing to noise on the channel the probability of receiving a transmitted '0' as a '1' is 1/16, while the probabi lity of transmitting a '1' and receiving a '0' is 1/32. Determine the rate at which information is received.**

Solution:

Rate of reception of information is given by –

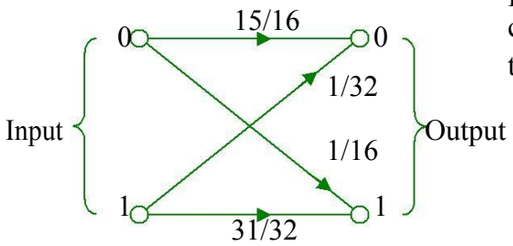
$R = H^1(X) - H^1(X/Y)$ bits / sec -----(1)

Where, $H(X) = -\sum_i p(i) \log p(i)$ bits / sym.

$H(X/Y) = -\sum_i \sum_j p(ij) \log p(i / j)$ bits / sym. -----(2)

$H(X) = -\left[\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right] = 1$ bit / sym.

Channel model or flow graph is



Index 'i' refers to the I/P of the channel and index 'j' refers to the output (R_X)

Probability of transmitting a symbol (i) given that a symbol '0' was received was received is denoted as p(i/j).

- $\sum_{j=0}^{\quad} \quad \sum_{i=0}^{\quad}$?
- **How would you compute p(0/0)**

  Recall the probability of a joint event AB p(AB)

  P(AB) = p(A) p(B/A) = p(B) p(A/B)

  i.e., p(ij) = p(i) p(j/i) = p(j) p(i/j)

  from which we have,

$$p(i/j) = \frac{p(i)\, p(j/i)}{p(j)} \qquad \text{-----(3)}$$

- **What are the different combinations of i & j in the present case?**

  Say i = 0 and j = 0, then equation (3) is p(0/0) $\frac{p(0)\, p(0/0)}{p(j=0)}$

- **What do you mean by p(j = 0)? And how to compute this quantity?**

  Substituting, find p(0/0)

  Thus, we have, $p(0/0) = \frac{p(0/0)\, p(0/0)}{p(0)}$

$$= \frac{\frac{1}{2} \times \frac{15}{16}}{\frac{31}{64}} = \frac{30}{31} = 0.967$$

$\therefore\ p(0/0) = 0.967$

- **Similarly calculate and check the following.**

$$p_0 = \frac{1}{31}, \quad p_1 = \frac{31}{33}; \quad p_0 = \frac{22}{33}$$

- **Calculate the entropy H(X/Y)**

$$H(X/Y) = -\ p(00)\log p\left(\frac{0}{0}\right) + p(01)\log p\left(\frac{0}{1}\right) + p(10)\log p\left(\frac{1}{0}\right) + p(11)\log p\left(\frac{1}{1}\right)$$

Substituting for the various probabilities we get,

$$H(X/Y) = -\frac{15}{32}\log\frac{30}{31} + \frac{1}{32}\log\frac{2}{33} + \frac{31}{64}\log p\ \frac{31}{33} + \frac{1}{64}\log\frac{1}{31}$$

Simplifying you get,

H(X/Y) = 0.27 bit/sym.

$\therefore$ [H(X) – H(X/Y)] . r_s

= (1 – 0.27) x 1000

$\therefore$ R = 730 bits/sec.

2. A transmitter produces three symbols ABC which are related with joint probability shown.

p(i)	**i**
9/27	**A**
16/27	**B**
2/27	**C**

p(j/i)		**j**		
		A	**B**	**C**
i	**A**	**0**	$\frac{4}{5}$	$\frac{1}{5}$
	B	$\frac{1}{2}$	$\frac{1}{2}$	**0**
	C	$\frac{1}{2}$	$\frac{2}{5}$	$\frac{1}{10}$

Calculate H(XY)

Solution:

By definition we have

H(XY) = H(X) + H(Y/X) -----(1)

Where, $H(X) = -\sum_i p(i)\log p(i)$ bits / symbol -----(2)

and $H(Y/X) = -\sum_i \sum_j p(ij)\log p(j/i)$ bits / symbol -----(3)

From equation (2) calculate H(X)

$\therefore$ H(X) = 1.257 bits/sym.

- To compute H(Y/X), first construct the p(ij) matrix using, p(ij) = p(i), p(j/i)

p(i, j)		j		
		A	**B**	**C**
i	**A**	**0**	$\frac{4}{15}$	$\frac{1}{15}$
	B	$\frac{8}{27}$	$\frac{8}{27}$	**0**
	C	$\frac{1}{27}$	$\frac{4}{135}$	$\frac{1}{135}$

- $\therefore$ **From equation (3), calculate H(Y/X) and verify, it is**
  H(Y/X) = 0.934 bits / sym.
- **Using equation (1) calculate H(XY)**
  $\therefore$ **H(XY) = H(X) + H(Y/X)**
  = 1.25 + 0.934

2.5 Capacity of a Discrete Memory less Channel (DMC):

- **Capacity of noisy**
  DMC Is Defined as –
  The maximum possible rate of information transmission over the channel.

In equation form –

$$C = \underset{P(x)}{\text{Max}} \left[D_t\right] \qquad \text{-----(1)}$$

i.e., maximized over a set of input probabilities P(x) for the discrete

Definition of D_t?

D_t: Ave. rate of information transmission over the channel defined as

$$Dt \triangleq \left[H(x) - H(x/y)\right] r_s \quad \text{bits / sec.} \qquad \text{-----(2)}$$

$\therefore$ Eqn. (1) becomes

$$C = \underset{P(x)}{\text{Max}} \left\{\left[H(x) - H(x/y)\right] r_s\right\} \qquad \text{-----(3)}$$

- What type of channel is this?
- Write the channel matrix

P(Y/X)		Y		
		0	?	1
X	0	p	q	p
	1	o	q	p

- Do you notice something special in this channel?
- What is H(x) for this channel?
  Say P(x=0) = P & P(x=1) = Q = (1 – P)
  H(x) = – P log P – Q log Q = – P log P – (1 – P) lo g (1 – P)
- What is H(y/x)?
  H(y/x) = – [p log p + q logq]

DISCRETE CHANNELS WITH MEMORY:

In such channels occurrence of error during a particular symbol interval does not influence the occurrence of errors during succeeding symbol intervals

– **No Inter-symbol Influence**

This will not be so in practical channels – Errors do not occur as independent events but tend to occur as **bursts.** Such channels are said to have Memory.

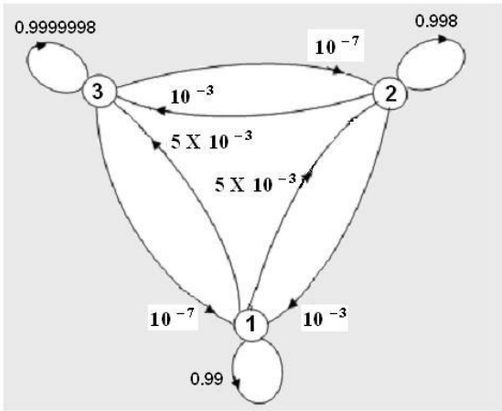
Examples:

– Telephone channels that are affected by switching transients and dropouts
– Microwave radio links that are subjected to fading

In these channels, **impulse noise** occasionally dominates the Gaussian noise and errors occur in infrequent long bursts.

Because of the complex physical phenomena involved, detailed characterization of channels with memory is very difficult.

GILBERT model is a model that has been moderately successful in characterizing error bursts in such channels. Here the channel is modeled as a discrete memoryless BSC, where the probability of error is a time varying parameter. The changes in probability of error are modeled by a Markoff process shown in the Fig 1 below.



State	1	2	3
Probability of Error	0.5	10^{-2}	10^{-6}

Fig 1. A Three - state Gilbert Model for communication channels

The error generating mechanism in the channel occupies one of three states.
Transition from one state to another is modeled by a discrete, stationary Mark off process.

For example, when the channel is in State 2 bit error probability during a bit interval is 10^{-2} and the channel stays in this state during the succeeding bit interval with a probability of 0.998.
However, the channel may go to state 1wherein the bit error probability is 0.5. Since the system stays in this state with probability of 0.99, errors tend to occur in bursts (or groups).
State 3 represents a low bit error rate, and errors in this state are produced by Gaussian noise. Errors very rarely occur in bursts while the channel is in this state.
Other details of the model are shown in Fig 1.
The maximum rate at which data can be sent over the channel can be computed for each state of the channel using the BSC model of the channel corresponding to each of the three states.
Other characteristic parameters of the channel such as the mean time between error bursts and mean duration of the error bursts can be calculated from the model.

2. LOGARITHMIC INEQUALITIES:

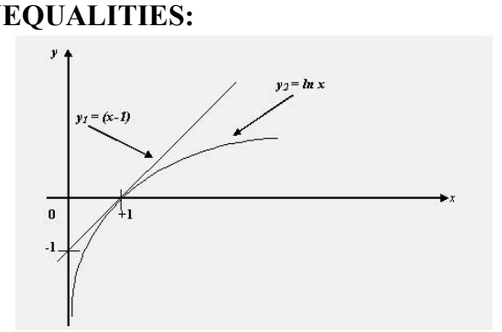


Fig 2. Plots of Straight line $y_1 = (x - 1)$ and $y_2 = \ln x$

Fig 2 shows the graphs of two functions $y_1 = x - 1$ and $y_2 = \ln x$ .The first function is a linear measure and the second function is your logarithmic measure. Observe that the log function always lies below the linear function , except at $x = 1$. Further the straight line is a tangent to the log function at $x = 1$.This is true only for the ***natural logarithms***. For example, $y_2 = \log_2 x$ is equal to $y_1 = x - 1$ *at* two points. Viz. at $x = 1$ and at x = *2* .In between these two values $y_1 > y_2$.*You should keep this point in mind when using the inequalities that are obtained.* From the graphs shown, it follows that, $y_1 \leq y_2$; *equality holds good if and only if* $x = 1$.*In other words:*

$$\ln x \leq (x-1), \quad \text{equality iff } x = 1 \qquad \ldots\ldots \quad (2.1)$$

Multiplying equation (2.1) throughout by '**-1**' and

noting that $-\ln x = \ln (1/x)$, we obtain another inequality as below.

This $\boxed{\ln \frac{1}{x} \geq (1-x),\ \text{equality iffy } x = 1}$ ……..…… (2.2)

property of the logarithmic function will be used in establishing the extremal property of the Entropy function (i.e. Maxima or minima property). As an additional property, let $\{p_1, p_2, p_3, \ldots..p_n\}$ and $\{q_1, q_2, q_3, \ldots..q_n\}$ be any two sets of probabilities such that $p_i \geq 0$, $q_j \geq 0$; $\forall$ i,j and $\sum_{i=1}^{n} p_i = \sum_{j=1}^{n} q_j$. Then we have:

$$\sum_{i=1}^{n} p_i \log_2 \frac{q_i}{p_i} = \log_2 e \cdot \sum_{i=1}^{n} p_i \ln \frac{q_i}{p_i}$$

Now using Eq (2.1), it follows that:

$$\sum_{i=1}^{n} p_i \log_2 \frac{q_i}{p_i} \leq \log_2 e \cdot \sum_{i=1}^{n} \frac{q_i}{p_i} - 1 \ ,\forall i = 1,2,...n$$

$$\leq \log_2 e \cdot \sum_{i=1}^{n} [q_i] - \sum_{i=1}^{n} [p_i],$$

$$\sum_{i=1}^{n} \frac{q_i}{p_i} \leq 0$$

That is, $\sum_{i=1}^{n} p_i \frac{1}{p_i} \leq \sum_{i=1}^{n} \frac{1}{q_i}$ equality iffy $p_i = q_i$ $\forall i=1, 2, 3...n$ ..... (2.3)

This inequality will be used later in arriving at a measure for code efficiency

3. PROPERTIES OF ENTROPY:

We shall now investigate the properties of entropy function

1. The entropy function is continuous in each and every independent variable ,p_k in the interval (0,1)

   This property follows since each p_k is continuous in the interval (***0, 1***) and that the logarithm of a continuous function is continuous by itself.

2. The entropy function is a symmetrical function of its arguments; i.e.

   $H(p_k, 1-p_k) = H(1-p_k, p_k)$ $\forall$ $k = 1, 2, 3...q.$

   That is to say that the value of the function remains same irrespective of the locations (positions) of the probabilities. In other words, as long as the probabilities of the set are same, it does not matter in which order they are arranged. Thus the sources S_1, S_2 and S_3 with probabilities:

$P_1 = \{p_1, p_2, p_3, p_4\}$, $P_2 = \{p_3, p_2, p_4, p_1\}$ and $P_3 = \{p_4, p_1, p_3, p_2\}$ with $\sum_{k=1}^{4} p_k = 1$ all have same entropy, i.e. $H(S_1) = H(S_2) = H(S_3)$

3. Extremal property:

Consider a zero memory information source with a q-symbol alphabet $S = \{s_1, s_2, s_3, \ldots, s_q\}$ with associated probabilities $P = \{p_1, p_2, p_3, \ldots, p_q\}$.

Then we have for the entropy of the source (as you have studied earlier):

$$H(S) = \sum_{k=1}^{q} p_k \log \frac{1}{p_k}$$

Consider $\log q - H(S)$. We have:

$$\begin{aligned}
\log q - H(S) &= \log q - \sum_{k=1}^{q} p_k \log \frac{1}{p_k} \\
&= \sum_{k=1}^{q} p_k \log q - \sum_{k=1}^{q} p_k \log \frac{1}{p_k} \\
&= \log e \sum_{k=1}^{q} p_k \ln q - \sum_{k=1}^{q} p_k \ln \frac{1}{p_k} \\
&= \log e \cdot \sum_{k=1}^{q} p_k \left(\ln q - \ln \frac{1}{p_k} \right) \\
&= \log e \cdot \sum_{k=1}^{q} p_k \left(\ln q\, p_k \right)
\end{aligned}$$

Invoking the inequality in Eq (2.2), we have:

$$\log q - H(S) \geq \log e \cdot \sum_{k=1}^{q} p_k \left(\frac{1}{q\, p_k} - 1 \right) \quad \text{Equality iff } q\, p_k = 1, \ \forall \ k = 1, 2, 3 \ldots q.$$

$$\geq \log e \left(\sum_{k=1}^{q} p_k - \sum_{k=1}^{q} \frac{1}{q} \right) \quad \text{Equality iff } p_k = 1/q, \ \forall \ k = 1, 2, 3 \ldots q.$$

Since $\sum_{k=1}^{q} p_k = \sum_{k=1}^{q} \frac{1}{q} = 1$, it follows that $\log q - H(S) \geq 0$

Or in other words $H(S) \leq \log q$ (2.4)

The equality holds good *iff* $p_k = 1/q$, $\forall$ $k = 1, 2, 3 \ldots q$. **Thus "for a zero memory information source, with a q-symbol alphabet, the Entropy becomes a maximum if and only if all the source symbols are equally probable".** *From Eq (2.4) it follows that:*

$H(S)_{max} = \log q$ … *iff* $p_k = 1/q$, $\forall$ $k = 1, 2, 3 \ldots q$ …….. (2.5)

Particular case- Zero memory binary sources:

For such a source, the source alphabet is $S= \{0, 1\}$ with $P= \{q, p\}$. Since $p+q=1$, we have $\lim_{p \to 0} p \log p = 0$, we define $H(S) = p \log (1/p) + q \log (1/q) = -p \log p-(1-p \log (1-p)$.Further, as $0 \log 0=0$ and $1 \log 1=0$. A sketch showing the variation of $H(S)$ with p is shown in Fig.2.5

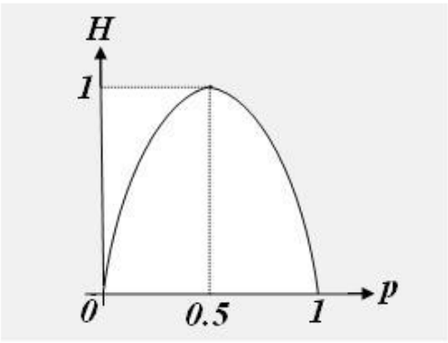


Fig 3. Entropy of a Binary Source

If the output of the source is certain (i.e. $p=0$ or $p =1$) then the source provides no information. The maximum entropy provided by the source is $\log_2 2 =1$ bit/binits, occurs iffy the ' 0' and the ' 1' are equally probable. The outputs of a binary source are 'Binary digits' or 'Binits'. Hence, a sequence of binits from a zero memory binary source with equi-probable 0's and 1's will provide 1 bit of information per binit. If the 0's and 1's are not equi-probable, then the amount of information provided by a given binit will be either less than or greater than 1 **bit**, depending upon its probability. However the 'average amount of information' provide d by a binit from such a source will always be less than or equal to 1 **bit per binit**.

EXTENSION OF A ZERO MEMORY SOURCE:

The questions of extension of sources arise in coding problems. If multi-alphabet source outputs are to be encoded in to words of a lesser alphabet, then it is necessary to have an extension of the later. For example , if we are to code four messages with a binary source $S=\{ 0 , 1 \}$, it is necessary to have the binary word $\{ 00 , 01 , 10 , 11 ,\}$, leading to the second extension of the source. Thus, if a zero memory source S has the alphabets $\{ s_1 , s_2 , , s_3 , \ldots, s_q \}$, then its n^{th} extension, called S^n, is a zero memory source with q^n symbols $\{ \sigma_1 ,\sigma_2 , \sigma_3 , \sigma_4 , \ldots , \sigma_{q^n} \}$ as its higher order alphabet. The corresponding statistics of the extension are given by the probabilities:

$$P(\sigma_i) = P(s_{i1}, s_{i2}, s_{i3} \ldots , s_{in}) \qquad \textbf{(Problem 2.1.6 Simon Haykin)}$$

where $\sigma_i = \{ s_{i1} , s_{i2} , s_{i3} \ldots , s_{in} \}$, that is , each σ_i corresponds to some sequence of n - symbols, s_i, of the source. Since it is a zero memory source all the resultant composite symbols are independent. There fore:

$P(\sigma_i) = P(s_{i1}).P(s_{i2}) .P(s_{i3}) \ldots P(s_{in})$. Further:

The condition that $\sum_{i=1}^{q^n} P(\sigma_i) = 1$ is satisfied , sin ce ,

$$\sum_{S^n} P(\sigma_i) = \sum_{i=1}^{q^n} \{P(s_{i1}).P(s_{i2}).....P(s_{in})\}$$

$$= \sum_{i_1=1}^{q} P(s_{i1}).\sum_{i_2=1}^{q} P(s_{i2}) ..... \sum_{i_n=1}^{q} P(s_{in})$$

$$= 1 \qquad \text{sin ce each } \sum_{q}(.) = 1$$

It then follows that:

$$H(S^n) = \sum_{S^n} P(\sigma_i).\log \frac{1}{P(\sigma_i)}$$

$$= \sum_{S^n} P(\sigma_i).\log \frac{1}{P(s_{i1})P(s_{i2}).P(s_{i3}).....P(s_{in})} .$$

$$= \sum_{S^n} P(\sigma_i).\log \frac{1}{P(s_{i1})} . + \sum_{S^n} P(\sigma_i).\log \frac{1}{P(s_{i2})} . + ...\sum_{S^n} P(\sigma_i).\log \frac{1}{P(s_{in})} .$$

Consider:

$$\sum_{S^n} P(\sigma_i).\log \frac{1}{P(s_{ij})} = \sum_{S^n} P(s_{i1}).P(s_{i2}). \quad ..... P(s_{in}) \log \frac{1}{P(s_{ij})}$$

$$= \sum_{i1=1}^{q} P(s_{i1}). \sum_{i2=1}^{q} P(s_{i2}). \quad ..... \sum_{ij=1}^{q} P(s_{ij}).\log \frac{1}{P(s_{ij})} . \quad .... \sum_{in=1}^{q} P(s_{in}).$$

$$= 1\,1 ..... \quad H(S) \quad .....1$$

$$= H(S)$$

$$Using\ this, and\ the\ fact\ that\ there\ are\ 'n'\ such\ terms : H(S^n) = n.H(S) \ ..... \qquad (2.6)$$

Thus the Entropy of the extension, $\boldsymbol{S^n}$*, of the Zero memory is 'n' times the Entropy of the original source.*

Example:

A zero memory source has a source alphabet S = $\{s_1, s_2, s_3\}$ *, with P =* ***{1/2, 1/4, 1/4}****. Find the entropy of the source. Find the entropy of the second extension and verify Eq (2.6).*
We have ***H(S) = (1/2) ×log (2) + 2× (1/4) × log 4 = 1.5 bits /sym***. The second extension and the corresponding probabilities are tabulated as below:

$$S^2 = \{s_1s_1 , s_1s_2 , s_1s_3 , s_2s_1 , s_2s_2 , s_2s_3 , s_3s_1 , s_3s_2 , s_3s_3 \}$$

$$P(S^2) = \{ 1/4 , 1/8 , 1/8 , 1/8 , 1/16 , 1/16 , 1/8 , 1/16 , 1/16 \}$$

Hence, $H(S^2) = (1/4) \times \log 4 + 4 \times (1/8) \log 8 + 4 \times (1/16) \times \log 16 = 3.0$ ***bits / sym***.
$\therefore \{H(S^2)\}/\{H(S)\} = 3 / 1.5 = 2$ ***; and indeed*** $H(S^2) = 2.H(S)$

SHANNON'S FIRST THEOREM (Noiseless Coding Theorem)**:**

" ***Given a code with an alphabet of r-symbols and a source with an alphabet of q-symbols, the average length of the code words per source symbol may be made as arbitrarily close to the lower bound H(S)/log r as desired by encoding extensions of the source rather than encoding each source symbol individually***".

The draw back is the increased coding complexity of the encoding procedure caused by the large number ($\boldsymbol{q^n}$) of source symbol with which we must deal and in the increased time required for encoding and transmitting the signals. Although the theorem has been proved here for zero memory sources, it is also valid for sources with memory i.e. for Markov sources.

Construction of some Basic Codes:

So far we have discussed the properties of the codes, bounds on the word lengths and the Shannon's first fundamental theorem. In this section we present some code generating techniques – Shannon's encoding procedure, Shannon – Fano codes and Huffman's minimum redundancy codes.

Shannon binary encoding procedure:

We present, first, the Shannon's procedure for generating binary codes mainly because of its historical significance. The procedure is based on Eq (5.32). The technique is easy to use and generates fairly efficient codes. The procedure is as follows:

1. List the source symbols in the order of decreasing probability of occurrence.

$S= \{s_1, s_2 \ldots s_q\}$; $P= \{p_1, p_2 \ldots .p_q\}: p_1 \geq p_2 \geq \ldots\ldots \geq p_q$

2. Compute the sequence:

$\alpha_0 = 0,$
$\alpha_1 = p_1,$
$\alpha_2 = p_2+p_1 = p_2+\alpha_1$
$\alpha_3=p_3+p_2+p_1 = p_3 + \alpha_2$
.
$\alpha_{q-1}= p_{q-1}+ p_{q-2}+ \ldots\ldots+p_1 = p_{q-1} + \alpha_{q-2}.$
$\alpha_q=p_q +p_{q-1}+ \ldots\ldots+p_1 = p_q + \alpha_{q-1} =1$

3. Determine the set of integers, l_k, which are the smallest integer's solution of the inequalities.
$2^{l_k} \; p_k \geq 1, k=1, 2, 3 \ldots q$. Or alternatively, find l_k such that $2^{-l_k} \leq p_k$.

4. Expand the decimal numbers α_k in binary form to l_k places. i.e., neglect expansion beyond l_k digits

5. Removal of the decimal point would result in the desired code.

Example 6.9:

Consider the following message ensemble
$S= \{s_1, s_2, s_3, s_4\}, P= \{0.4, 0.3, 0.2, 0.1\}$

Then following Shannon's procedure, we have

1) $0.4 > 0.3 > 0.2 > 0.1$

2) $\alpha_0 = 0,$
$\alpha_1 = 0.4$
$\alpha_2 = 0.4+0.3=0.7$
$\alpha_3= 0.7+0.2=0.9$
$\alpha_4= 0.9 + 0.1 = 1.0$

3) $2^{-l_1} \le 0.4 \rightarrow l_1 = 2$

$2^{-l_2} \le 0.3 \rightarrow l_2 = 2$

$2^{-l_3} \le 0.2 \rightarrow l_3 = 3$

$2^{-l_4} \le 0.1 \rightarrow l_4 = 4$

4) $\alpha_0 = 0 = 0.00\ \underline{0}$ α_1
$= 0.4 = 0.01\ \underline{10}$
$\alpha_2 = 0.7 = 0.101\ \underline{10}$
$\alpha_3 = 0.9 = 0.1110\ \underline{01}$

5) The codes are

$s_1 \rightarrow 00, s_2 \rightarrow 01, s_3 \rightarrow 101, s_4 \rightarrow 1110$

The average length of this code is

$L = 2 \times 0.4 + 2 \times 0.3 + 3 \times 0.2 + 3 \times 0.1 = 2.4$ Bini ts / message

$$H(S) = 0.4 \log \frac{1}{0.4} + 0.3 \log \frac{1}{0.3} + 0.2 \log \frac{1}{0.2} + 0.1 \log \frac{1}{0.1} = 1.84644 \text{ bits / message;}$$

$$\log 2 = 1.0 \text{ and } \eta_c = \frac{H(S)}{L \log r} = \frac{1.84644}{2.4 \times 1} = 76.935\ \%\ ;\ E_c = 23.065\ \%$$

Shannon – Fano Binary Encoding Method:

Shannon – Fano procedure is the simplest available. Code obtained will be optimum if and only if $p_k = r^{-l_k}$ .The procedure is as follows:

1. List the source symbols in the order of decreasing probabilities.
2. Partition this ensemble into almost two equi- probable groups.
   Assign a '*0*' to one group and a '*1*' to the other group. These form the starting code symbols of the codes.
3. Repeat steps *2* and *3* on each of the subgroups until the subgroups contain only one source symbol, to determine the succeeding code symbols of the code words.
4. For convenience, a code tree may be constructed and codes read off directly.

Example

Consider the message ensemble $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$ with

$$P = \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}\right\}, X = \{0, 1\}$$

The procedure is clearly indicated in the Box diagram shown in Fig 6.3. The Tree diagram for the steps followed is also shown in Fig 6.4. The codes obtained are also clearly shown. For this example,

$$L = 2\times\frac{1}{4} + 2\times\frac{1}{4} + 3\times\frac{1}{8} + 3\times\frac{1}{8} + 4\times\frac{1}{16} + 4\times\frac{1}{16} + 4\times\frac{1}{16} + 4\times\frac{1}{16} = 2.75 \text{ binits / symbol}$$

$$H(S) = 2\times\frac{1}{4}\log 4 + 2\times\frac{1}{8}\log 8 + 4\times\frac{1}{16}\log 16 = 2.75 \text{ bits/symbol.}$$

And as $\log r = \log 2 = 1$, we have $\eta_c = \dfrac{H(S)}{L \log r} = 100\%$ and $E_c = 0\%$

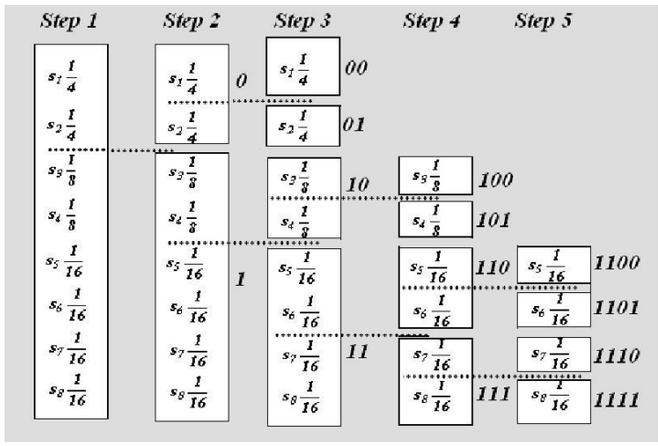


Fig 6.3 Box Diagram for Example 5.10

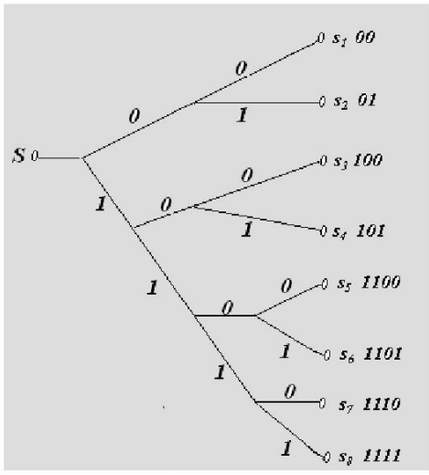


Fig 6.4 Tree diagram for Example 5.10

Incidentally, notice from tree diagram that the ***codes originate from the same source and diverge into different tree branches*** and hence it is clear that no complete code can be a prefix of any other code word. Thus the Shannon- Fano algorithm provides us a means for constructing optimum, instantaneous codes.

In making the partitions, remember that the symbol with highest probability should be made to correspond to a code with shortest word length. Consider the binary encoding of the following message ensemble.

Example 6.11

$S = \{s1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$

$P = \{0.4 , 0.2 , 0.12 , 0.08 , 0.08 , 0.08 , 0.04\}$

Method - I

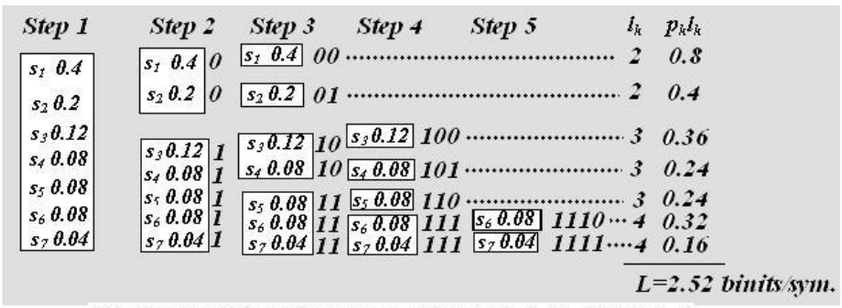


Fig 5.5 Partition diagram for Example 5.11 Method- I

Method - II

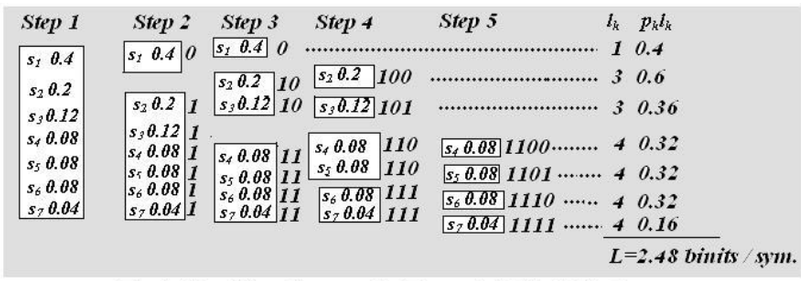


Fig 5.6 Partition diagram for Example 5.11 Method-II

For the partitions adopted, we find ***L=2.52 binits / sym***. for the ***Method – I***

L=2.48 binits/sym for the ***Method - II***

For this example, ***H(S) =2.420504 bits/sym*** and

For the first method, $\eta_{c1} = 96.052\%$

For the second method, $\eta_{c\,2} = 97.6\%$

This example clearly illustrates the logical reasoning required while making partitions. The Shannon – Fano algorithm just says that the message ensemble should be partitioned into two almost equi-probable groups. While making such partitions care should be taken to make sure that the symbol with highest probability of occurrence will get a code word of minimum possible length. In

the example illustrated, notice that even though both methods are dividing the message ensemble into two almost equi-probable groups, the Method – II as signs a code word of smallest possible length to the symbol s_1.

Review questions :

1. What do you mean by source encoding? Name the functional requirements to be satisfied in the development of an efficient source encoder.

2. For a binary communication system, a '0' or '1' is transmitted. Because of noise on the channel, a '0' can be received as '1' and vice-versa. Let m_0 and m_1 represent the events of transmitting '0' and '1' respectively. Let r_0 and r_0 denote the events of receiving '0' and '1' respect ively. Let $p(m_0) = 0.5$, $p(r_1/m_0) = p = 0.1$, $P(r_0/m_1) = q = 0.2$
   i. Find $p(r_0)$ and $p(r_1)$
   ii. If a '0' was received what is the probability that '0' was sent
   iii. If a '1' was received what is the probability that '1' was sent.
   iv. Calculate the probability of error.
   v. Calculate the probability that the transmitted symbol is read correctly at the receiver.

3. State Shannon-Hartley's law. Derive an equation showing the efficiency of a system in terms of the information rate per Unit bandwidth. How is the efficiency of the system related to B/W?

4. For a discrete memory less source of entropy H(S), show that the average code-word length for any distortion less source encoding scheme is bounded as $L \geq H(S)$.

5. Calculate the capacity of a standard 4KHz telephone channel working in the range of 200 to 3300 KHz with a S/N ratio of 30 dB.

6. What is the meaning of the term communication channel. Briefly explain data communication channel, coding channel and modulation channel.

7. Obtain the communication capacity of a noiseless channel transmitting n discrete message system/sec.

8. Explain extremal property and additivity property.

9. Suppose that S1, S2 are two memory sources with probabilities p1,p2,p3,……pn for source s1 and q1,q2,…….qn for source s2 . Show that the entro py of the source s1

$$H(s1) \leq \sum_{K=1}^{n} Pk \log (1/qk)$$

10. Explain the concept of B/W and S/N trade-off with reference to the communication channel.

.

Unit – 3 : Fundamental Limits on Performance

Syllabus :

Source coding theorem, Huffman coding, Discrete memory less Channels, Mutual information, Channel Capacity. **6 Hours**

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996. Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007

Unit – 3 : Fundamental Limits on Performance

Source coding theorem:
Compact code: Huffman's Minimum Redundancy code:

The Huffman code was created by American, D. A. Huffman, in 1952. Huffman`s procedure is applicable for both Binary and Non- Binary encoding. It is clear that a code with minimum average length, L, would be more efficient and hence would have minimum redundancy associated with it. A compact code is one which achieves this objective. Thus for an optimum coding we require:

1) Longer code word should correspond to a message with lowest probability.

2) $l_k \geq l_{k-1} \; \forall \; k = 1,2,.....q^{-r+1}$

(3) $l_{p-r} = l_{q-r-1} = l_{q-r-2} = \ldots.. = l_q$

(4) The codes must satisfy the prefix property.

Huffman has suggested a simple method that guarantees an optimal code even if Eq. (6.13) is not satisfied. The procedure consists of step- by- step reduction of the original source followed by a code construction, starting with the final reduced source and working backwards to the original source. The procedure requires α steps, where

$$q = r + \alpha(r-1) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6.24)$$

Notice that α is an integer and if Eq.(6.24) is not satisfied one has to add few dummy symbols with zero probability of occurrence and proceed with the procedure or the first step is performed by setting $r_1 = q-\alpha(r-1)$ while the remaining steps involve clubbing of the last r messages of the respective stages. The procedure is as follows:

1. List the source symbols in the decreasing order of probabilities.

2. Check if $q = r + \alpha(r-1)$ is satisfied and find the integer ' α' . Otherwise add suitable number of dummy symbols of zero probability of occurrence to satisfy the equation. ***This step is not required if we are to determine binary codes.***

3. Club the last r symbols into a single composite symbol whose probability of occurrence is equal to the sum of the probabilities of occurrence of the last r – symbols involved in the step.

4. Repeat steps *1* and *3* respectively on the resulting set of symbols until in the final step exactly r- symbols are left.

5. Assign codes freely to the last r-composite symbols and work backwards to the original source to arrive at the optimum code.

6. Alternatively, following the steps carefully a tree diagram can be constructed starting from the final step and codes read off directly.

7. Discard the codes of the dummy symbols.

Before we present an example, it is in order to discuss the steps involved. In the first step, after arranging the symbols in the decreasing order of probabilities; we club the last ***r***-symbols into a composite symbol, say σ_1 whose probability equals the sum of the last ***r***-probabilities. Now we are left with ***q-r+1*** symbols .In the second step, we again club the last ***r***-symbols and the second reduced source will now have (***q-r+1***)-***r***+***1***= ***q-2r+2*** symbols .Continuing in this way we find the ***k***-th reduced source will have ***q- kr + k = q – k***(***r - 1***) symbols. Accordingly, if α -steps are required and the final reduced source should have exactly ***r***-symbols, then we must have ***r = q - α (r - 1)*** and Eq (5.38) is proved. However, notice that if Eq (6.24) is not satisfied, we may just start the first step by taking the last $r_1 = q - \alpha(r-1)$ symbols while the second and subsequent reductions involve last ***r***-symbols only. However, if the reader has any confusion, he can add the dummy messages as indicated and continue with the procedure and the final result is no different at all.

Let us understand the meaning of " *working backwards*". Suppose α_k is the composite symbol obtained in the k^{th} step by clubbing the last ***r***-Symbols of the $(k-1)^{th}$ reduced source. ***Then whatever code is assigned to α_k will form the starting code sequence for the code words of its constituents in the $(k-1)^{th}$ reduction.***

Example 6.12: (*Binary Encoding*)

$$S = \{s_1, s_2, s_3, s_4, s_5, s_6\},\ X = \{0, 1\};$$

$$P = \frac{1}{3}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{12}, \frac{1}{12}$$

The reduction diagram and the tree diagram are given in Fig 5.7.Notice that the tree diagram can be easily constructed from the final step of the source reduction and decomposing the composite symbols towards the original symbols. Further, observe that the codes are originating from the same source and diverge out into different tree branches thus ensuring prefix property to the resultant code .Finally, notice that there is no restriction in the allocation of codes in each step and accordingly, the order of the assignment can be changed in any or all steps. Thus for the problem illustrated there can be as many as ***2. (2.2+2.2) = 16*** possible instantaneous code patterns. For example we can take the compliments of First column, Second column, or Third column and combinations there off as illustrated below.

	Code	*I*	*II*	*III*
s_1	0 0	1 0	1 1	1 1
s_2	1 0	0 0	0 1	0 1
s_3	0 1 0	1 1 0	1 0 0	1 0 1
s_4	0 1 1	1 1 1	1 0 1	1 0 0
s_5	1 1 0	0 1 0	0 0 0	0 0 1
s_6	1 1 1	0 1 1	0 0 1	0 0 0

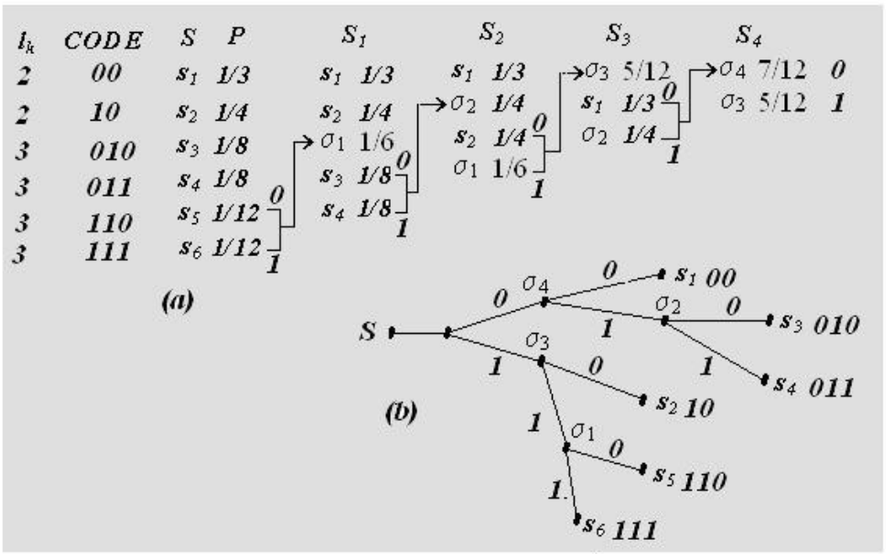


Fig 5.7 (a) Reduction Diagram (b) Tree Diagram For Binary Huffman Coding of Example 5.12

Code I is obtained by taking complement of the first column of the original code. ***Code II*** is obtained by taking complements of second column of ***Code I***. ***Code III*** is obtained by taking complements of third column of ***Code II***. However, notice that, l_k, the word length of the code word for s_k is a constant for all possible codes.

For the binary code generated, we have:

$$L = \sum_{k=1}^{6} p_k l_k = \frac{1}{3} \times 2 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 + \frac{1}{12} \times 3 + \frac{1}{12} \times 3 = \frac{29}{12} \text{ binits/sym} = 2.4167 \text{ binits/sym}$$

$$H(S) = \frac{1}{3} \log 3 + \frac{1}{4} \log 4 + 2 \times \frac{1}{8} \log 8 + 2 \times \frac{1}{12} \log 12$$

$$= \frac{1}{12}(6 \log 3 + 19) \text{ bits/sym} = 2.3758 \text{ bits/ sym}$$

$$\therefore \eta_c = \frac{6 \log 3 + 19}{29} = 98.31\% \; ; E_c = 1.69\%$$

Example 6.13 (*Trinary Coding*)

We shall consider the source of Example 6.12. For Trinary codes $r = 3$, $[X = (0, 1, 2)]$
Since $q = 6$, we have from

$$q = r + \alpha(r-1)$$

$$\alpha = \frac{q - r}{r - 1} = \frac{6 - 3}{2} = \frac{3}{2} = 1.5$$

Thus α is not an integer and hence *we require one dummy message which makes* $\alpha = 2$.

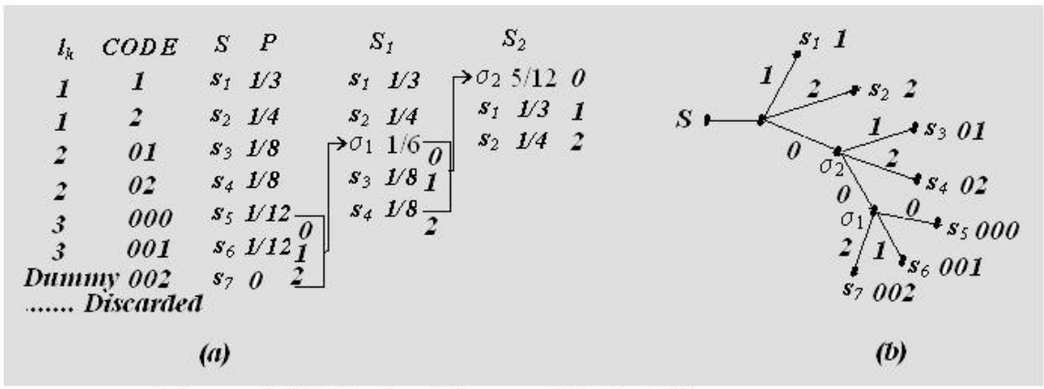


Fig 5.8 (a) Reduction Diagram (b) Tree Diagram
For Trinary Huffman Coding of Example 5.13

For this code, we have

$$L = 1\times\frac{1}{3} + 1\times\frac{1}{4} + 2\times\frac{1}{8} + 2\times\frac{1}{8} + 3\times\frac{1}{12} + 3\times\frac{1}{12} = \frac{19}{12} \text{ Trinits / sym .}$$

And $\eta_c = \dfrac{6\log 3 + 19}{19} = 94.672\%$, $E_c = 5.328\%$

Example 6.14:

We conclude this section with an example illustrating Shannon's noiseless coding theorem. Consider a source $S = \{s_1, s, s_3\}$ with $P = \{1/2, 1/3, 1/6\}$

A compact code for this source is: $s_1 \rightarrow 0$, $s_2 \rightarrow 10$, $s_3 \rightarrow 11$

Hence we have

$$L = \frac{1}{2} + \frac{2}{3} + \frac{2}{6} = 1.5$$

$$H(S) = \frac{1}{2}\log 2 + \frac{1}{3}\log 3 + \frac{1}{6}\log 6$$

$$= 1.459147917 \text{ bits/sym}$$

$$\therefore \eta_c = 97.28\%$$

The second extension of this source will have $3^2 = 9$ symbols and the corresponding probabilities are computed by multiplying the constituent probabilities as shown below

$s_1 s_1$	$\frac{1}{4}$	$s_2 s_1$	$\frac{1}{6}$	$s_3 s_1$	$\frac{1}{12}$
$s_1 s_2$	$\frac{1}{6}$	$s_2 s_2$	$\frac{1}{9}$	$s_3 s_2$	$\frac{1}{18}$
$s_1 s_3$	$\frac{1}{12}$	$s_2 s_3$	$\frac{1}{18}$	$s_3 s_3$	$\frac{1}{36}$

These messages are now labeled ' m_k' and are arranged in the decreasing order of probability.

$M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9\}$

$$P = \frac{1}{4}, \frac{1}{6}, \frac{1}{6}, \frac{1}{9}, \frac{1}{12}, \frac{1}{12}, \frac{1}{18}, \frac{1}{18}, \frac{1}{36}$$

The Reduction diagram and tree diagram for code construction of the second extended source is shown in Fig 5.9.

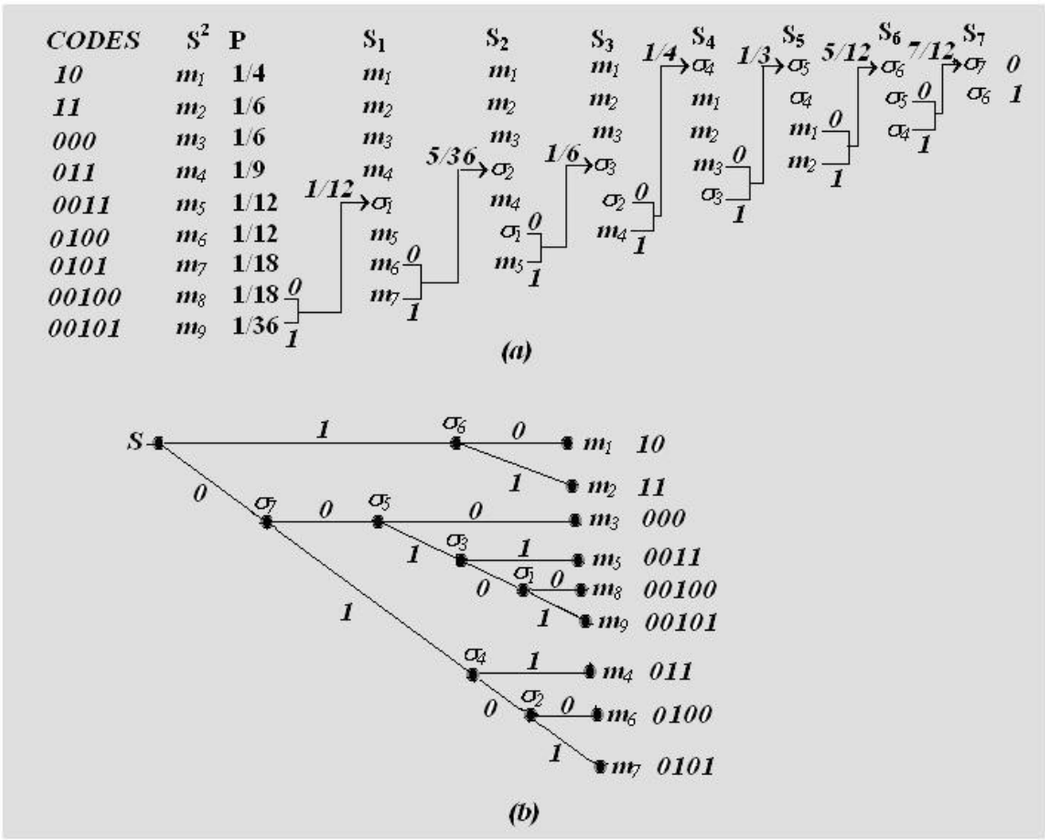


Fig 5.9 (a) Reduction Diagram (b) Tree Diagram for Example 5.14

For the codes of second extension, we have the following:

$$H(S^2) = 2H(S)$$

$$L = 2\times\frac{1}{4} + 2\times\frac{1}{6} + 3\times\frac{1}{6} + 3\times\frac{1}{9} + 4\times\frac{1}{12} + 4\times\frac{1}{12} + 4\times\frac{1}{18} + 5\times\frac{1}{18} + 5\times\frac{1}{36}$$

$$= \frac{107}{36} \text{ binits/symbol} = 2.97222222 \text{ binits/sym}$$

$$\eta_c = \frac{H(S^2)}{L \log 2} = \frac{2 \times 1.459147917}{2.97222222} = 98.186\% \quad E_c = 1.814\%$$

An increase in efficiency of ***0.909 %*** (absolute) is achieved.

This problem illustrates how encoding of extensions increase the efficiency of coding in accordance with Shannon's noiseless coding theorem.

One non- uniqueness in Huffman coding arises in making decisions as to where to move a composite symbol when you come across identical probabilities. In Shannon- Fano binary encoding you came across a situation where you are required to make a logical reasoning in deciding the partitioning. To illustrate this point, consider the following example.

Example 6.15:

Consider a zero memory source with

S= {s_1, s_2, s_3, s_4, s_5}; P= {0.55, 0.15, 0.15, 0.10, 0.05}; X= {0, 1}

Construct two different Huffman binary codes as directed below:

(a) Move the composite symbol as 'high' as possible.

(b) *Move the composite symbol as 'low' as possible*

(c) *In each case compute the variance of the word lengths and comment on the results.*

(a)We shall ***place*** the composite symbol as ' ***high***' as possible. The source reduction and the corresponding tree diagram are shown in Fig 6.10

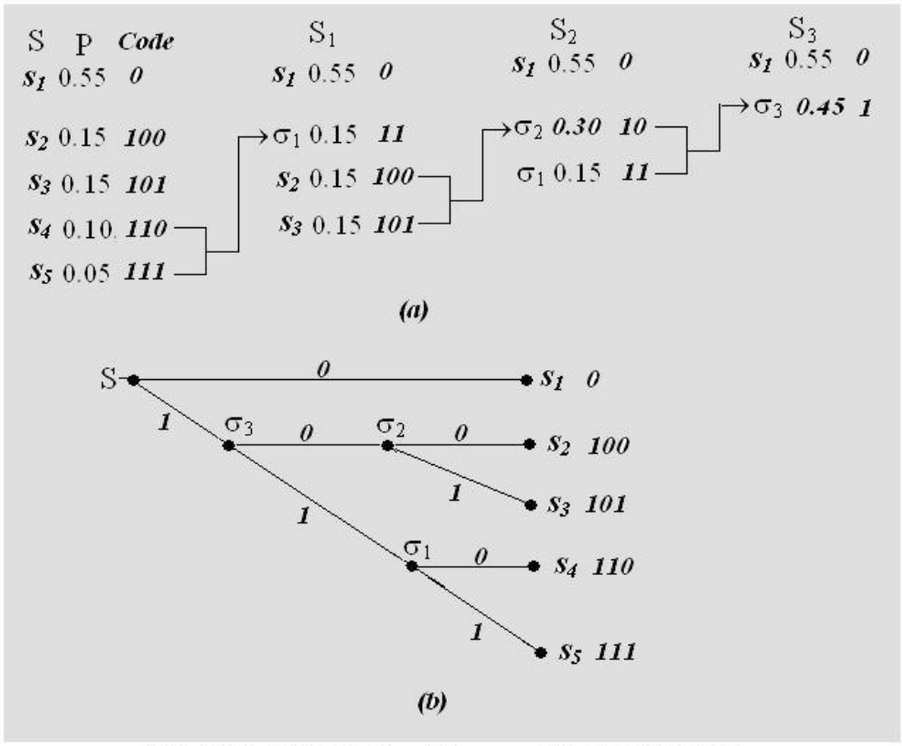


Fig 5.10 (a) Reduction Diagram (b) Tree Diagram

Symbols	***s_1***	***s_2***	***s_3***	***s_4***	***s_5***
Codes	***0***	***100***	***101***	***110***	***111***
l_k	***1***	***3***	***3***	***3***	***3***

We compute the average word length and variance of the word lengths as below:

L=0.55+3(0.15+0.15+0.10+0.05) =1.90 binits/symbol

σ^2_l ***= 0.55(1-1.90)^2 + 0.45 (3-19)^2 = 0.99*** is the variance of the word length.

(a) We shall ***move*** the composite symbol as ' ***low***' as possible. The source reduction and the corresponding tree diagram are shown in Fig 5.11.We get yet another code, completely different in structure to the previous one.

Symbols	***s_1***	***s_2***	***s_3***	***s_4***	***s_5***

Codes	*0*	*11*	*100*	*1010*	*1011*
l_k	*1*	*2*	*3*	*4*	*4*

For this case we have: ***L = 0.55 + 0.30 + 0.45 + 0.20= 1.90 binits/symbol***

Notice that the average length of the codes is same.

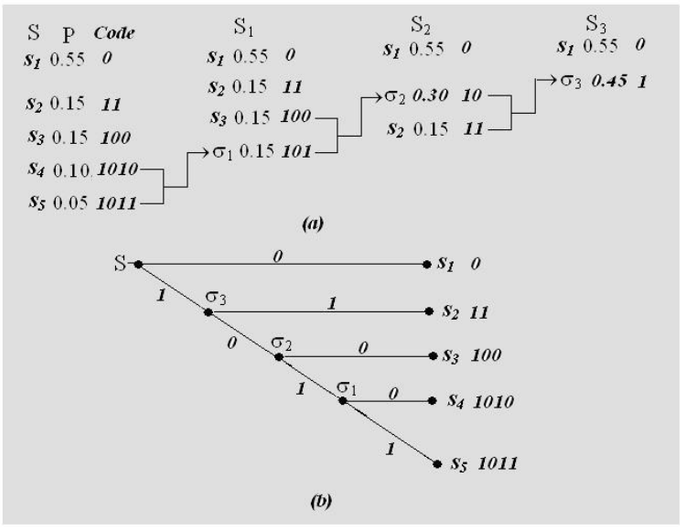


Fig 5.11 (a) Reduction Diagram (b) Tree Diagram

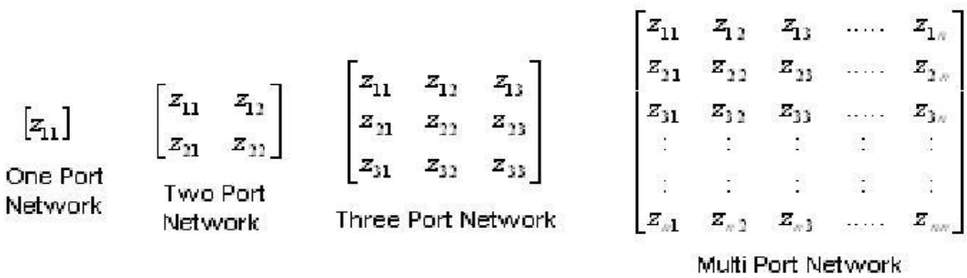
$$\sigma^2_2 = 0.55\,(1-1.9)^2 + 0.15\,(2-1.9)^2 + 0.15(3-1.9)^2 + 0.10(4-1.9)^2 + 0.05(4-1.9)^2$$

$= 1.29$ is the variance of the word lengths.

Thus, if the composite symbol is moved as high as possible, the variance of the average code word length over the ensemble of source symbols would become smaller, which, indeed, is desirable. Larger variance implies larger buffer requirement for storage purposes. Further, if the variance is large, there is always a possibility of data overflow and the time required to transmit information would be larger. We must avoid such a situation. Hence we always look for codes that have minimum possible variance of the word lengths. Intuitively " *avoid reducing a reduced symbol in the immediate next step as far as possible moving the composite symbol as high as possible*".

DISCRETE MEMORYLESS CHANNELS:

A multi-port electric network may be uniquely described by the impedance matrices, viz,



Observe that the matrix is necessarily a square matrix. The principal diagonal entries are the self impedances of the respective ports. The off diagonal entries correspond to the transfer or mutual impedances. For a passive network the impedance matrix is always symmetric i.e. $Z^T = Z$, where the superscript indicates transposition.

Similarly, a communication network may be uniquely described by specifying the joint probabilities (JPM). Let us consider a simple communication network comprising of a transmitter (source or input) and a receiver (sink or output) with the interlinking medium-the channel as shown in Fig 4.1.

Fig 4.1 A Simple Communication System

This simple system may be uniquely characterized by the ' **Joint probability matrix**' (**JPM**), ***P(X, Y)*** of the probabilities existent between the input and output ports.

$$\begin{matrix} p(x_1,y_1) & p(x_1,y_2) & P(x_1,y_3) & ..... & P(x_1,y_n) \\ P(x_2,y_1) & P(x_2,y_2) & P(x_2,y_3) & ..... & P(x_2,y_n) \\ P(x_3,y_1) & P(x_3,y_2) & P(x_3,y_3) & ..... & P(x_3,y_n) \\ M & M & M & M & M \\ M & M & M & M & M \\ P(x_m,y_1) & P(x_m,y_2) & P(x_m,y_3) & ..... & P(x_m,y_n) \end{matrix} \quad ..... \quad (4.1)$$

For jointly continuous random variables, the joint density function satisfies the following:

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)dxdy = 1 \quad ..... \quad (4.2)$$

$$\int_{-\infty}^{+\infty} f(x,y)dy = f_X(x) \quad ..... \quad (4.3)$$

$$\int_{-\infty}^{+\infty} f(x,y)dx = f_Y(y) \quad ..... \quad (4.4)$$

We shall make use of their discrete counterpart as below:

$$\sum_{\forall k}\sum_{\forall j} p(x_k,y_j) = 1 \text{ ,Sum of all entries of } \textbf{JPM} \quad ..... \quad (4.5)$$

$$\sum_{\forall j} p(x_k,y_j) = p(x_k) \text{ , Sum of all entries of } \textbf{JPM} \text{ in the k}^{th} \text{ r} \quad .....(4.6)$$

$$\sum_{\forall k} p(x_k,y_j) = p(y_j) \text{ , Sum of all entries of } \textbf{JPM} \text{ in the j}^{th} \text{ column} \quad ..... \quad (4.7)$$

And also

$$\sum_{\forall k} p(x_k) = \sum_{\forall j} p(y_j) = 1 \qquad \text{.....} \qquad \text{.....} \qquad (4.8)$$

Thus the joint probabilities, as also the conditional probabilities (as we shall see shortly) form complete finite schemes. Therefore for this simple communication network there are five probability schemes of interest viz: ***P*(*X*)**, ***P*(*Y*)**, ***P*(*X*, *Y*)**, ***P* (*X*|*Y*)** and ***P* (*Y*|*X*)**. Accordingly there are five entropy functions that can be described on these probabilities:

H(X): Average information per character or symbol transmitted by the source or the entropy of the source.

H(Y): Average information received per character at the receiver or the entropy of the receiver.

H(X, Y): Average information per pair of transmitted and received characters or the average uncertainty of the communication system as a whole.

H (X|Y): A specific character yj being received. This may be the result of the transmission of one of the xk with a given probability. The average value of the Entropy associated with this scheme when yj covers all the received symbols i.e., E {H (X|yj)} is the entropy H (X|Y), called the 'Equivocation', a measure of information about the source when it is known that Y is received.

H (Y|X) : Similar to H (X|Y), this is a measure of information about the receiver.

The marginal Entropies ***H*(*X*)** and ***H*(*Y*)** give indications of the probabilistic nature of the transmitter and receiver respectively. ***H* (*Y*|*X*)** indicates a measure of the 'noise' or 'error' in t he channel and the equivocation ***H*(*X* |*Y*)** tells about the ability of recovery or reconstruction of the transmitted symbols from the observed output symbols.

The above idea can be generalized to an ***n***- port communication system, problem being similar to the study of random vectors in a product space (***n***-dimensional random variables Theory). In each product space there are finite numbers of probability assignments (joint, marginal and conditional) of different orders, with which we may associate entropies and arrive at suitable physical interpretation. However, concepts developed for a two-dimensional scheme will be sufficient to understand and generalize the results for a higher order communication system.

Joint and Conditional Entropies:

In view of Eq (4.2) to Eq (4.5), it is clear that all the probabilities encountered in a two dimensional communication system could be derived from the **JPM**. While we can compare the **JPM**, therefore, to the impedance or admittance matrices of an ***n***-port electric network in giving a unique description of the system under consideration, notice that the **JPM** in general, need not necessarily be a square matrix and even if it is so, it need not be symmetric.

We define the following entropies, which can be directly computed from the **JPM**.

$$H(X, Y) = p(x_1, y_1) \log \frac{1}{p(x_1, y_1)} + p(x_1, y_2) \log \frac{1}{p(x_1, y_2)} + \ldots + p(x_1, y_n) \log \frac{1}{p(x_1, y_n)}$$
$$+ p(x_2, y_1) \log \frac{1}{p(x_2, y_1)} + p(x_2, y_2) \log \frac{1}{p(x_2, y_2)} + \ldots + p(x_2, y_n) \log \frac{1}{p(x_1, y_1)}$$

$$+\ldots\ p(x_m, y_1)\log\frac{1}{p(x_m, y_1)} + p(x_m, y_2)\log\frac{1}{p(x_m, y_2)} + \ldots\ p(x_m, y_n)\log\frac{1}{p(x_m, y_n)} \text{ or}$$

$$H(X, Y) = \sum_{k=1}^{m}\sum_{j=1}^{n} p(x_k, y_j)\log\frac{1}{p(x_k, y_j)} \qquad \ldots\ldots\ldots\ldots\ldots(4.9)$$

$$H(X) = \sum_{k=1}^{m} p(x_k)\log\frac{1}{p(x_k)}$$

Using Eq (4.6) only for the multiplication term, this equation can be re-written as:

$$H(X) = \sum_{k=1}^{m}\sum_{j=1}^{n} p(x_k, y_j)\log\frac{1}{p(x_k)} \qquad \ldots\ldots\ldots\ldots \quad (4.10)$$

Similarly, $H(Y) = \sum_{j=1}^{n}\sum_{k=1}^{m} p(x_k, y_j)\log\frac{1}{p(y_j)}$ ..... (4.11)

Next, from the definition of the conditional probability we have:

$$P\{X = x_k \mid Y = y_j\} = \frac{P\{X = x_k, Y = y_j\}}{P\{Y = y_j\}}$$

i.e., $p(x_k \mid y_j) = p(x_k, y_j)\ ) / p(y_j)$

Then $\sum_{k=1}^{m} p(x_k \mid y_j) = \frac{1}{p(y_j)}\sum_{k=1}^{m} p(x_k, y_j) = \frac{1}{p(y_j)} \cdot p(y_j) = 1$ ..... (4.12)

Thus, the set $[X \mid y_j] = \{x_1 \mid y_j, x_2 \mid y_j \ldots x_m \mid y_j\}$; $P[X \mid y_j] = \{p(x_1 \mid y_j), p(x_2 \mid y_j) \ldots p(x_m \mid y_j)\}$, forms a complete finite scheme and an entropy function may therefore be defined for this scheme as below:

$$H(X \mid y_j) = \sum_{k=1}^{m} p(x_k \mid y_j)\log\frac{1}{p(x_k \mid y_j)}.$$

Taking the average of the above entropy function for all admissible characters received, we have the average " ***conditional Entropy" or "Equivocation":***

$$\begin{aligned} H(X \mid Y) &= E\{H(X \mid y_j)\}_j \\ &= \sum_{j=1}^{n} p(y_j)\, H(X \mid y_j) \\ &= \sum_{j=1}^{n} p(y_j)\sum_{k=1}^{m} p(x_k \mid y_j)\log\frac{1}{p(x_k \mid y_j)} \end{aligned}$$

Or $H(X \mid Y) = \sum_{j=1}^{nm}\sum_{k=1} p(x_k, y_j)\log\frac{1}{p(x_k \mid y_j)}$ ..... (4.13)

Eq (4.13) specifies the " ***Equivocation*** ". It specifies the average amount of information n eeded to specify an input character provided we are allowed to make an observation of the output produced by that input. Similarly one can define the conditional entropy $H(Y \mid X)$ by:

$$H(Y|X) = \sum_{k=1}^{m}\sum_{j=1}^{n} p(x_k, y_j) \log \frac{1}{p(y_j|x_k)} \quad \ldots\ldots\ldots\ldots \quad (4.14)$$

Observe that the manipulations, made in deriving Eq 4.10, Eq 4.11, Eq 4.13 and Eq 4.14, are intentional. ' ***The entropy you want is simply the double summation of joint probability multiplied by logarithm of the reciprocal of the probability of interest'*** . For example, if you want joint entropy, then the probability of interest will be joint probability. If you want source entropy, probability of interest will be the source probability. If you want the equivocation or conditional entropy, $H(X|Y)$ then probability of interest will be the conditional probability $p(x_K|y_j)$ and so on.

All the five entropies so defined are all inter-related. For example, consider Eq (4.14). We have:

$$H(Y|X) = \sum_k \sum_j p(x_k, y_j) \log \frac{1}{p(y_j|x_k)}$$

Since, $\frac{1}{p(y_j|x_k)} = \frac{p(x_k)}{p(x_k, y_j)}$

We can straight away write:

$$H(Y|X) = \sum_k \sum_j p(x_k, y_j) log \frac{1}{p(y_j|x_k)} - \sum_k \sum_j p(x_k, y_j) log \frac{1}{p(x_k)}$$

Or $H(Y|X) = H(X, Y) - H(X)$

That is: $H(X, Y) = H(X) + H(Y|X)$ ..... (4.15)

Similarly, you can show: $H(X, Y) = H(Y) + H(X|Y)$ ..... (4.16)

Consider $H(X) - H(X|Y)$. We have:

$$H(X) - H(X|Y) = \sum_k \sum_j p(x_k, y_j) \log \frac{1}{p(x_k)} - \log \frac{1}{p(x_k|y_j)}$$

$$= \sum_k \sum_j p(x_k, y_j) log \frac{p(x_k, y_j)}{p(x_k).p(y_j)} \quad \ldots\ldots \quad (4.17)$$

Using the logarithm inequality derived earlier, you can write the above equation as:

$$H(X) - H(X|Y) = log\ e \sum_k \sum_j p(x_k, y_j) ln \frac{p(x_k, y_j)}{p(x_k).p(y_j)}$$

$$\geq log\ e \sum_k \sum_j p(x_k, y_j) \left(1 - \frac{p(x_k).p(y_j)}{p(x_k, y_j)}\right)$$

$$\geq log\ e \left(\sum_k \sum_j p(x_k, y_j) - \sum_k \sum_j p(x_k).p(y_j)\right)$$

$$\geq log\ e \left(\sum_k \sum_j p(x_k, y_j) - \sum_k p(x_k).\sum_j p(y_j)\right) \geq 0$$

Because $\sum_k \sum_j p(x_k, y_j) = \sum_k p(x_k) = \sum_j p(y_j) = 1$. Thus it follows that:

$$H(X) \geq H(X|Y) \quad \ldots\ldots \quad (4.18)$$

Similarly, $H(Y) \geq H(Y|X)$ …….. (4. 19)

Equality in Eq (4.18) &Eq (4.19) holds iffy P $(x_k, y_j) = p(x_k).p(y_j)$; *i*.e., **if and only if input symbols and output symbols are statistically independent of each other.**

NOTE : Whenever you write the conditional probability matrices you should bear in mind the property described in Eq.(4.12), i.e. **For the CPM (conditional probability matrix) $P(X|Y)$, if you add all the elements in any column the sum shall be equal to unity. Similarly, if you add all elements along any row of the CPM, $P(Y|X)$ the sum shall be unity**

Example 4.1

Determine different entropies for the JPM given below and verify their relationships.

$$P(X, Y) = \begin{matrix} 0.2 & 0 & 0.2 & 0 \\ 0.1 & 0.01 & 0.01 & 0.01 \\ 0 & 0.02 & 0.02 & 0 \\ 0.04 & 0.04 & 0.01 & 0.06 \\ 0 & 0.06 & 0.02 & 0.2 \end{matrix}$$

Using $p(x_k) = \sum_{j=1}^{n} p(x_k, y_j)$, we have, by adding entries of $P(X, Y)$ row-wise we get:

$$P(X) = [0.4, 0.1, 0.04, 0.15, 0.28]$$

Similarly adding the entries column-wise we get:

$$P(Y) = [0.34, 0.13, 0.26, 0.27]$$

Hence we have:

$$H(X,Y) = 3 \times 0.2 \log \frac{1}{0.2} + 0.1 \times \log \frac{1}{0.1} + 4 \times 0.01 \log \frac{1}{0.01} + 3 \times 0.02 \log \frac{1}{0.02} + 2 \times 0.04 \log \frac{1}{0.04} + 2 \times 0.06 \log \frac{1}{0.06}$$

$$= 3.188311023 \text{ bits /sym}$$

$$H(X) = 0.4 \log \frac{1}{0.4} + 0.13 \log \frac{1}{0.13} + 0.04 \log \frac{1}{0.04} + 0.15 \log \frac{1}{0.15} + 0.28 \log \frac{1}{0.28}$$

$$= 2.021934821 \text{ bits / sym}$$

$$H(Y) = 0.34 \log \frac{1}{0.34} + 0.13 \log \frac{1}{0.13} + 0.26 \log \frac{1}{0.26} + 0.27 \log \frac{1}{0.27}$$

$$= 1.927127708 \text{ bits / sym}$$

Since $p(x_k | y_j) = \frac{P(x_k, y_j)}{P(y_j)}$ we have:

(Divide the entries in the j^{th} column of the **JPM** of $p(y_j)$

$$P(X|Y) = \begin{matrix} \frac{0.2}{0.34} & 0 & \frac{0.2}{0.26} & 0 \\ \frac{0.1}{0.34} & \frac{0.01}{0.13} & \frac{0.01}{0.26} & \frac{0.01}{0.27} \\ 0 & \frac{0.02}{0.13} & \frac{0.02}{0.26} & 0 \\ \frac{0.04}{0.34} & \frac{0.04}{0.13} & \frac{0.01}{0.26} & \frac{0.06}{0.27} \\ 0 & \frac{0.06}{0.13} & \frac{0.02}{0.26} & \frac{0.20}{0.27} \end{matrix}$$

$$\therefore H(X|Y) = 0.2\log\frac{0.34}{0.2} + 0.2\log\frac{0.26}{0.2} + 0.1\log\frac{0.34}{0.1}$$
$$+ 0.01\log\frac{0.13}{0.01} + 0.01\log\frac{0.26}{0.01} + 0.01\log\frac{0.27}{0.01}$$
$$+ 0.02\log\frac{0.13}{0.02} + 0.02\log\frac{0.26}{0.02} + 0.04\log\frac{0.34}{0.04}$$
$$+ 0.04\log\frac{0.13}{0.04} + 0.01\log\frac{0.26}{0.01} + 0.06\log\frac{0.27}{0.06}$$
$$+ 0.06\log\frac{0.13}{0.06} + 0.02\log\frac{0.26}{0.02} + 0.2\log\frac{0.27}{0.2}$$

$$= 1.261183315 \text{ bits / symbol}$$

Similarly, dividing the entries in the k^{th} row of **JPM** by $p(x_k)$, we obtain the **CPM** $P(Y|X)$.Then we have:

$$P(Y|X) = \begin{matrix} \frac{0.2}{0.4} & 0 & \frac{0.2}{0.4} & 0 \\ \frac{0.1}{0.13} & \frac{0.01}{0.13} & \frac{0.01}{0.13} & \frac{0.01}{0.13} \\ 0 & \frac{0.02}{0.04} & \frac{0.02}{0.04} & 0 \\ \frac{0.04}{0.15} & \frac{0.04}{0.15} & \frac{0.01}{0.15} & \frac{0.06}{0.15} \\ 0 & \frac{0.06}{0.28} & \frac{0.02}{0.28} & \frac{0.20}{0.28} \end{matrix}$$

$$And\ \ H(Y|X) = 2\times 0.2\log\frac{0.4}{0.2} + 0.1\log\frac{0.13}{0.1} + 3\times 0.01\log\frac{0.13}{0.01} + 2\times 0.02\log\frac{0.04}{0.02}$$
$$+ 2\times 0.04\log\frac{0.05}{0.04} + 0.01\log\frac{0.15}{0.01} + 0.06\log\frac{0.15}{0.06} + 0.06\log\frac{0.28}{0.06}$$
$$+ 2\times 0.02\log\frac{0.28}{0.02} = 1.166376202\ bits/sym\ .$$

Thus by actual computation we have

***H*(*X*, *Y*) = 3.188311023 bits/Sym *H*(*X*)= 2.02193482 bit/Sym *H*(*Y*)= 1.927127708 bits/Sym**

***H*(*X* | *Y*) = 1.261183315 bits/Sym *H*(*Y* | *X*) = 1.166376202 bits/Sym**

Clearly, ***H*(*X*, *Y*) = *H*(*X*) + *H*(*Y* | *X*) = *H*(*Y*) + *H*(*X* | *Y*)**

***H*(*X*) > *H*(*X* | *Y*)** and ***H*(*Y*) > *H*(*Y* | *X*)**

Mutual information:

On an average we require ***H*(*X*)** bits of information to specify one input symbol. However, if we are allowed to observe the output symbol produced by that input, we require, then, only ***H* (*X*|*Y*)** bits of information to specify the input symbol. Accordingly, we come to the conclusion, that on an average, observation of a single output provides with [***H*(*X*) – *H* (*X*|*Y*)**] bits of information. This difference is called ' ***Mutual Information***' or ' ***Transinformation***' of the channel, denoted by ***I*(*X*, *Y*).** Thus:

$$I(X, Y) \triangleq H(X) - H(X|Y) \qquad \text{..... (4.20)}$$

Notice that in spite of the variations in the source probabilities, $p(x_k)$ (may be due to noise in the channel), certain probabilistic information regarding the state of the input is available, once the conditional probability $p(x_k \mid y_j)$ is computed at the receiver end. The difference between the initial uncertainty of the source symbol x_k, i.e. $\log 1/p(x_k)$ and the final uncertainty about the same source symbol x_k, after receiving y_j, **i.e.** $\log 1/p(x_k \mid y_j)$ is the information gained through the channel. This difference we call as the mutual information between the symbols x_k and y_j. Thus

$$I(x_k, y_j) = \log\frac{1}{p(x_k)} - \log\frac{1}{p(x_k \mid y_j)}$$
$$= \log\frac{p(x_k \mid y_j)}{p(x_k)} \qquad \text{.....(4.21 } a)$$

$$\textbf{Or } I(x_k, y_j) = \log\frac{p(x_k, y_j)}{p(x_k).p(y_j)} \qquad \text{..... (4.21 } b)$$

Notice from Eq. (4.21*a*) that

$$I(x_k) = I(x_k, x_k) = \log\frac{p(x_k \mid x_k)}{p(x_k)} = \log\frac{1}{p(x_k)}$$

This is the definition with which we started our discussion on information theory! Accordingly $I(x_k)$ is also referred to as 'Self Information'.

It is clear from Eq (3.21*b*) that, as $\frac{p(x_k, y_j)}{p(x_k)} = p(y_j \mid x_k)$,

$$I(x_k, y_j) = \log \frac{p(y_j \mid x_k)}{p(y_j)} = \log \frac{1}{p(y_j)} - \log \frac{1}{p(y_j \mid x_k)}$$

Or $$I(x_k, y_j) = I(y_j) - I(y_j \mid x_k) \quad \text{..... (4.22)}$$

Eq (4.22) simply means that "the Mutual information ' is symmetrical with respect to its arguments.i.e.

$$I(x_k, y_j) = I(y_j, x_k) \quad \text{..... (4.23)}$$

Averaging Eq. (4.21*b*) over all admissible characters x_k and y_j, we obtain the average information gain of the receiver:

$$I(X, Y) = E\{I(x_k, y_j)\} = \sum_k \sum_j I(x_k, y_j).\, p(x_k, y_j)$$

$$= \sum_k \sum_j p(x_k, y_j).\log \frac{p(x_k, y_j)}{p(x_k)p(y_j)} \quad \text{..... (4.24)}$$

From Eq (4.24) we have:

$$1)\ I(X, Y) = \sum_k \sum_j p(x_k, y_j)\ \log \frac{1}{p(x_k)} - \log \frac{1}{p(x_k \mid y_j)} = H(X) - H(X \mid Y)$$

..... (4.25)

$$2)\quad I(X, Y) = \sum_k \sum_j p(x_k, y_j)\ [\log \frac{1}{p(y_j)} - \log \frac{1}{p(y_j \mid x_k)}$$

$$= H(Y) - H(Y \mid X) \quad \text{..... (4. 26)}$$

$$3)\ I(X,Y) = \sum_k \sum_j p(x_k, y_j) \log \frac{1}{p(x_k)} + \sum_k \sum_j p(x_k, y_j)\ \log \frac{1}{p(y_j)} -$$

$$\sum_K \sum_J p(x_k, y_j) \log \frac{1}{p(x_k\ y_j)}$$

Or $$I(X, Y) = H(X) + H(Y) - H(X, Y) \quad \text{..... (4.27)}$$

Further, in view of Eq.(4.18) & Eq.(4.19) we conclude that, " ***even though for a particular received symbol, y_j, H(X) – H(X | Yj) may be negative, when all the admissible ou tput symbols are covered, the average mutual information is always non- negative***". That is to say, we cannot loose information on an average by observing the output of a channel. An easy method, of remembering the various relationships, is given in Fig 4.2.Althogh the diagram resembles a Venn-diagram, it is not, and the diagram is only a tool to remember the relationships. That is all. You cannot use this diagram for proving any result.

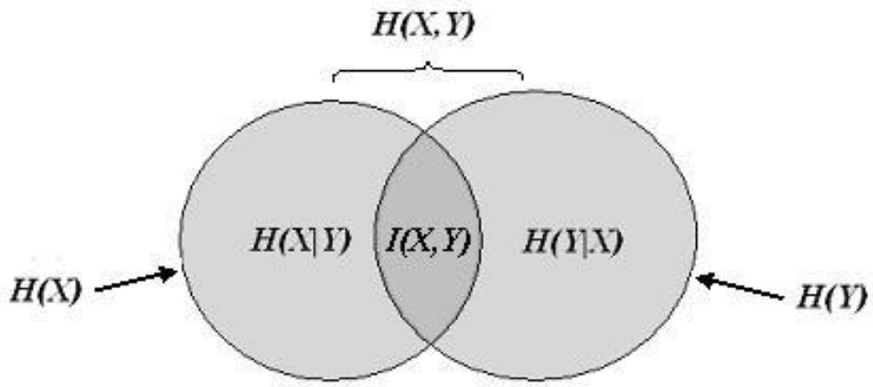


Fig 4.2 Entropy Relations

The entropy of ***X*** is represented by the circle on the left and that of ***Y*** by the circle on the right. The overlap between the two circles (dark gray) is the mutual information so that the remaining (light gray) portions of ***H*(*X*)** and ***H*(*Y*)** represent respective equivocations. Thus we have

$$H(X \mid Y) = H(X) - I(X, Y) \text{ and } H(Y \mid X) = H(Y) - I(X, Y)$$

The joint entropy ***H*(*X*,*Y*)** is the sum of ***H*(*X*)** and ***H*(*Y*)** except for the fact that the overlap is added twice so that

$$H(X, Y) = H(X) + H(Y) - I(X, Y)$$

Also observe $H(X, Y) = H(X) + H(Y|X)$
$= H(Y) + H(X|Y)$

For the **JPM** given in Example 4.1, $I(X, Y) = 0.760751505$ **bits / sym**

Shannon Theorem: Channel Capacity:

Clearly, the mutual information ***I (X, Y)*** depends on the source probabilities apart from the channel probabilities. For a general information channel we can always make ***I(X, Y) = 0*** by choosing any one of the input symbols with a probability one or by choosing a channel with independent input and output. Since ***I(X, Y)*** is always nonnegative, we thus know the minimum value of the Transinformation. However, the question of max ***I(X, Y)*** for a general channel is not easily answered.

Our intention is to introduce a suitable measure for the efficiency of the channel by making a comparison between the actual rate and the upper bound on the rate of transmission of information. Shannon's contribution in this respect is most significant. Without botheration about the proof, let us see what this contribution is.

Shannon's theorem: on channel capacity("coding Theo rem")

It is possible, in principle, to device a means where by a communication system will transmit information with an arbitrary small probability of error, provided that the information rate ***R***(=$r \times I$(***X***,***Y***),where ***r*** is the symbol rate) is less than or equal to a rate ' ***C***" called "channel capacity".

The technique used to achieve this objective is called coding. To put the matter more formally, the theorem is split into two parts and we have the following statements.

Positive statement:

" *Given a source of **M** equally likely messages, with **M>>1**, which is generating information at a rate **R**, and a channel with a capacity **C**. If **R ≤ C**, then there exists a coding technique such that the output of the source may be transmitted with a probability of error of receiving the message that can be made arbitrarily small*".

This theorem indicates that for ***R≤ C*** transmission may be accomplished without error even in the presence of noise. The situation is analogous to an electric circuit that comprises of only pure capacitors and pure inductors. In such a circuit there is no loss of energy at all as the reactors have the property of storing energy rather than dissipating.

Negative statement:

" *Given the source of **M** equally likely messages with **M>>1**, which is generating information at a rate **R** and a channel with capacity **C**. Then, if **R>C**, then the probability of error of receiving the message is close to unity for every set of **M** transmitted symbols*".

This theorem shows that if the information rate ***R*** exceeds a specified value ***C***, the error probability will increase towards unity as ***M*** increases. Also, in general, increase in the complexity of the coding results in an increase in the probability of error. Notice that the situation is analogous to an electric network that is made up of pure resistors. In such a circuit, whatever energy is supplied, it will be dissipated in the form of heat and thus is a "lossy network".

You can interpret in this way: Information is poured in to your communication channel. You should receive this without any loss. Situation is similar to pouring water into a tumbler. Once the tumbler is full, further pouring results in an over flow. You cannot pour water more than your tumbler can hold. Over flow is the loss.

Shannon defines " ***C***" the channel capacity of a communication channel a s the maximum value of Transinformation, ***I(X, Y)*:**

$$C = \Delta Max\ I(X, Y) = Max\ [H(X) - H(Y|X)] \quad \text{..... (4.28)}$$

The maximization in Eq (4.28) is with respect to all possible sets of probabilities that could be assigned to the input symbols. Recall the maximum power transfer theorem: 'In any network, maximum power will be delivered to the load only when the load and the source are properly matched'. The device used for this matching purpose, we shall call a "transducer ". For example, in a radio receiver, for optimum response, the impedance of the loud speaker will be matched to the impedance of the output power amplifier, through an output transformer.

This theorem is also known as "The Channel Coding Theorem" (Noisy Coding Theorem). It may be stated in a different form as below:

$$R \le C \text{ or } r_s H(S) \le r_c I(X,Y)_{Max} \text{ or } \{H(S)/T_s\} \le \{I(X,Y)_{Max}/T_c\}$$

"If a discrete memoryless source with an alphabet 'S' has an entropy H(S) and produces symbols every 'T $_s$' seconds; and a discrete memoryless channel has a capacity $I(X,Y)_{Max}$ and is used once every T_c seconds; then if

There exists a coding scheme for which the source output can be transmitted over the channel and be reconstructed with an arbitrarily small probability of error. The parameter C/T_C is called the critical rate. When this condition is satisfied with the equality sign, the system is said to be signaling at the critical rate.

Conversely, if $\frac{H(S)}{T_S} > \frac{I(X,Y)_{Max}}{T_C}$ ***, it is not possible to transmit information over the channel and reconstruct it with an arbitrarily small probability of error***

A communication channel, is more frequently, described by specifying the source probabilities ***P(X)*** & the conditional probabilities ***P (Y|X)*** rather than specifying the **JPM**. The **CPM**, ***P (Y|X),*** is usually refereed to as the ' ***noise characteristic***' of the channel. Therefore unless otherwise specified, we shall understand that the description of the channel, by a matrix or by a 'Channel diagram' always refers to **CPM,** ***P (Y|X)***. Thus, in a discrete communication channel with pre-specified noise characteristics (i.e. with a given transition probability matrix, ***P (Y|X))*** the rate of information transmission depends on the source that drives the channel. Then, the maximum rate corresponds to a proper matching of the source and the channel. This ideal characterization of the source depends in turn on the transition probability characteristics of the given channel.

Redundancy and Efficiency:

A redundant source is one that produces 'dependent' symbols. (Example: The Markov source). Such a source generates symbols that are not absolutely essential to convey information. As an illustration, let us consider the English language. It is really unnecessary to write "U" following the letter "Q". The redundancy in English text is e stimated to be 50%(refer J Das etal, Sham Shanmugam, Reza, Abramson, Hancock for detailed discussion.) This implies that, in the long run, half the symbols are unnecessary! For example, consider the following sentence.

" ***Y.u sh..ld b. abl. t. re.d t.is ev.n tho… sev.r.l l.t..rs .r. m.s..ng*** "

However, we want redundancy. Without this redundancy abbreviations would be impossible and any two dimensional array of letters would form a crossword puzzle! We want redundancy even in communications to facilitate error detection and error correction. Then how to measure redundancy? Recall that for a Markov source, ***H(S) < H($\bar{S}$)***, where ***$\bar{S}$*** is an ad- joint, zero memory source. That is, when dependence creeps in, the entropy of the source will be reduced and this can be used as a measure indeed!

" ***The redundancy of a sequence of symbols is measured by noting the amount by which the entropy has been reduced***".

When there is no inter symbol influence the entropy at the receiver would be ***H(X)*** for any given set of messages {***X***} and that when inter symbol influence occurs the entropy would be ***H (Y|X)***. The difference [***H(X) –H (Y|X)***] is the net reduction in entropy and is called " ***Absolute Redundancy***". Generally it is measured relative to the maximum entropy and thus we have for the " ***Relative Redundancy***" or simply, ' ***redundancy*** ' ***, E***

E = (Absolute Redundancy) ÷ H(X)

Or $$E = 1 - \frac{H(Y|X)}{H(X)} \quad \text{..... (4.29)}$$

Careful observation of the statements made above leads to the following alternative definition for redundancy,

$$E = 1 - \frac{R}{C} \quad \text{..... (4.30)}$$

Where R is the actual rate of Transinformation (mutual information) and C is the channel capacity. From the above discussions, a definition for the efficiency, η for the channel immediately follows:

$$\eta = \frac{\text{Actual rate of mutual information}}{\text{maximum possible rate}}$$

That is. $$\eta = \frac{R}{C} \quad \text{..... (4.31)}$$

and $$\eta = 1 - E \quad \text{..... (4.32)}$$

Capacity of Channels:

While commenting on the definition of 'Channel capacity', Eq. (4.28), we have said that maximization should be with respect to all possible sets of input symbol probabilities. Accordingly, to arrive at the maximum value it is necessary to use some Calculus of Variation techniques and the problem, in general, is quite involved.

Example 3.2: Consider a Binary channel specified by the following noise characteristic (channel matrix):

$$P(Y|X) = \begin{matrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{3}{4} \end{matrix}$$

The source probabilities are: $p(x_1) = p, p(x_2) = q = 1-p$

Clearly, $H(X) = -p \log p - (1 - p) \log (1 - p)$

We shall first find **JPM** and proceed as below:

$$P(X,Y) = \begin{matrix} p(x_1).p(y_1|x_1) & p(x_1).p(y2|x_1) \\ p(x2).p(y_1|x_2) & p(x2).p(y_2|x_2) \end{matrix} = \begin{matrix} \frac{p}{2} & \frac{p}{2} \\ \frac{1-p}{4} & \frac{3(1-p)}{4} \end{matrix}$$

Adding column-wise, we get:

$$p(y_1) = \frac{p}{2} + \frac{1-p}{4} = \frac{1+p}{4} \quad \text{and} \quad p(y_2) = \frac{p}{2} + \frac{3(1-p)}{4} = \frac{3-p}{4}$$

Hence $H(Y) = \frac{1+p}{4}\log\frac{4}{1+p} + \frac{3-p}{4}\log\frac{4}{3-p}$

And $H(Y|X) = \frac{p}{2}\log 2 + \frac{p}{2}\log 2 + \frac{1-p}{4}\log 4 + \frac{3(1-p)}{4}\log\frac{4}{3}$

$$I(X,Y) = H(Y) - H(Y|X) = 1 - \frac{3\log 3}{4} p + \frac{3\log 3}{4} - \frac{1+p}{4}\log(1+p) - \frac{3(1-p)}{4}\log(3-p)$$

Writing $\log x = \log e \times \ln x$ and setting $\frac{dI}{dp} = 0$ yields straight away:

$p = \frac{3a-1}{1+a} = 0.488372093$, Where $a = 2^{(4-3\log 3)} = 0.592592593$

With this value of p, we find $I(X,Y)_{Max} = 0.048821$ bits /sym

For other values of p it is seen that $I(X,Y)$ is less than $I(X,Y)_{max}$

Although, we have solved the problem in a straight forward way, it will not be the case when the dimension of the channel matrix is more than two. We have thus shown that the channel capacity of a given channel indeed depends on the source probabilities. The computation of the channel capacity would become simpler for certain class of channels called the 'symmetric 'or 'uniform' channels.

p	.0.2	.0.4	.0.5	.0.6	.0.8
I(X,Y) Bits / sym	.0.32268399	.0.04730118	.0.04879494	.0.046439344	.0.030518829

Muroga's Theorem :

The channel capacity of a channel whose noise characteristic, $P(Y|X)$, is square and non-singular, the channel capacity is given by the equation:

$$C = \log \sum_{i=1}^{i=n} 2^{-Q_i} \qquad \text{.....(4.33)}$$

Where Q_i are the solutions of the matrix equation $P(Y|X).Q = [h]$, where $h = [h_1, h_2, h_3, h_4 \ldots h_n]^t$ are the row entropies of $P(Y|X)$.

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & \ldots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \ldots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & p_{n3} & \ldots & p_{nn} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix}$$

$$C = \log \sum_{i=1}^{n} 2^{-Q_i}$$

$p_i' = 2^{-Q_i - C}$, where p_i' are obtained from: $p_i' = p_1 \cdot p_{1i} + p_2 \cdot p_{2i} + p_3 \cdot p_{3i} + \ldots + p_n \cdot p_{ni}$

$$[p_i]' = [p_1 \quad p_2 \quad \ldots\ldots \quad p_n] \begin{bmatrix} p_{1i} \\ p_{2i} \\ \vdots \\ p_{ni} \end{bmatrix}$$

$\downarrow$

i-th column of $P(Y|X)$

Or $[p_1', p_2', p_3' \ldots p_n'] = [p_1, p_2, p_3 \ldots p_n]\, P(Y|X)$.

From this we can solve for the source probabilities (i.e. Input symbol probabilities):

$[p_1, p_2, p_3 \ldots p_n] = [p_1', p_2', p_3' \ldots p_n']\, P^{-1}(Y|X)$, provided the inverse exists.

However, although the method provides us with the correct answer for Channel capacity, this value of C may not necessarily lead to physically realizable values of probabilities and if $P^{-1}(Y|X)$ does not exist ,we will not have a solution for Q_i`s as well. One reason is that we are not able to incorporate the inequality constraints $0 \leq p_i \leq 1$ .Still, within certain limits; the method is indeed very useful.

Example 4.2: Consider a Binary channel specified by the following noise characteristic (channel matrix):

$$P(Y|X) = \begin{matrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{3}{4} \end{matrix}$$

The row entropies are:

$$h_1 = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit/symbol}.$$

$$h_2 = \frac{1}{4}\log 4 + \frac{3}{4}\log \frac{4}{3} = 0.8112781 \text{ bits/symbol}.$$

$$P^{-1}(Y|X) = \begin{matrix} 3 & -1 \\ -1 & 2 \end{matrix}$$

$$Q_1 = P^{-1}(Y|X) \cdot \begin{matrix} h_1 \\ n_2 \end{matrix} = \begin{matrix} 1.3774438 \\ 0.6225562 \end{matrix}$$

$$C = \log\left[2^{-Q_1} + 2^{-Q_2}\right] = 0.048821 \text{ bits/symbol}, \quad \text{as before.}$$

$$\text{Further}, \quad p_1 = 2^{-Q_1 - C} = 0.372093 \quad \text{and} \quad p_2' = 2^{-Q_2 - C} = 0.627907.$$

$$[p_1 \quad p_2] = [p_1' \quad p_2'] \cdot P^{-1}(Y|X) = [0.488372 \quad 0.511628]$$

Giving us $p = 0.488372$

Example 4.3:

Consider a *3×3* channel matrix as below:

$$P[Y|X] = \begin{matrix} 0.4 & 0.6 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.6 & 0.4 \end{matrix}$$

The row entropies are:

$$h_1 = h_3 = 0.4 \log (1/0.4) + 0.6 \log (1/0.6) = 0.9709505 \text{ bits / symbol.}$$

$$h2 = 2 \times 0.5 \log (1/0.5) = 1 \text{ bit / symbol.}$$

$$P^{-1}[Y|X] = \begin{matrix} 1.25 & 1 & -1.25 \\ 5/6 & -2/3 & 5/6 \\ -1.25 & 1 & 1.25 \end{matrix}$$

$$\begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \begin{matrix} 1 \\ = 1.0193633 \\ 1 \end{matrix}$$

$$C = \log \{2^{-1} + 2^{-1.0193633} + 2^{-1}\} = 0.5785369 \text{ bits / symbol.}$$

$$p'_1 = 2^{-Q1 - C} = 0.3348213 = p'_3, \; p'_2 = 2^{-Q2 - C} = 0.3303574.$$

Therefore, $p_1 = p_3 = 0.2752978$ and $p2 = 0.4494043$.

Suppose we change the channel matrix to:

$$P[Y|X] = \begin{matrix} 0.8 & 0.2 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.2 & 0.8 \end{matrix} \qquad P^{-1}[Y|X] = \begin{matrix} 0.625 & 1 & -0.625 \\ 2.5 & -4 & 2.5 \\ -0.625 & 1 & 0.625 \end{matrix}$$

We have:

$h_1 = h_3 = 0.721928$ bits / symbol, and $h_2 = 1$ bit / symbol.

This results in:

$$Q_1 = Q_3 = 1; \; Q_2 = -0.39036.$$

$$C = \log \{2 \times 2^{-1} + 2^{+0.39036}\} = 1.2083427 \text{ bits / symbol.}$$

$$p'_1 = 2^{-Q1 - C} = 0.2163827 = p'_3, \; p'_2 = 2^{-Q2 - C} = 0.5672345$$

Giving: $p_1 = p_3 = 1.4180863$ and $p2 =$ Negative!

Thus we see that, although we get the answer for ***C*** the input symbol probabilities computed are not physically realizable. However, in the derivation of the equations, as already pointed out, had we included the conditions on both input and output probabilities we might have got an excellent result! But such a derivation becomes very formidable as you cannot arrive at a numerical solution! You will have to resolve your problem by graphical methods only which will also be a tough proposition! The formula can be used, however, with restrictions on the channel transition probabilities. For example, in the previous problem, for a physically realizable p_1, p_{11} should be less than or equal to ***0.64***. (Problems 4.16 and 4.18 of Sam Shanmugam to be solved using this method)

Symmetric Channels:

The Muroga's approach is useful only when the noise characteristic ***P [X|Y]*** is a square and invertible matrix. For channels with $m \neq n$, we can determine the Channel capacity by simple inspection when the channel is " ***Symmetric" or "Uniform".***

Consider a channel defined by the noise characteristic:

$$P[Y|X] = \begin{matrix} p_{11} & p_{12} & p_{13} & \dots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \dots & p_{2n} \\ p_{31} & p_{32} & p_{33} & \dots & p_{3n} \\ \text{M} & \text{M} & \text{M} & \text{M} & \text{M} \\ p_{n1} & p_{n2} & p_{n3} & \dots & p_{nn} \end{matrix} \quad \dots\dots\dots\dots\dots \quad (4.34)$$

This channel is said to be Symmetric or Uniform if the second and subsequent rows of the channel matrix are certain permutations of the first row. That is the elements of the second and subsequent rows are exactly the same as those of the first row except for their locations. This is illustrated by the following matrix:

$$P[Y|X] = \begin{matrix} p_1 & p_2 & p_3 & \dots & p_n \\ p_{n-1} & p_2 & p_n & \dots & p_4 \\ p_3 & p_2 & p_1 & \dots & p_5 \\ \text{M} & \text{M} & \text{M} & \text{M} & \text{M} \\ p_n & p_{n-1} & p_{n-2} & \dots & p_1 \end{matrix} \quad \dots\dots\dots\dots\dots \quad (4.35)$$

Remembering the important property of the conditional probability matrix, ***P*** [***Y|X***], that the sum of all elements in any row should add to unity; we have:

$$\sum_{j=1}^{n} p_j = 1 \quad \dots\dots\dots\dots\dots\dots \quad (4.36)$$

The conditional entropy ***H*** (***Y|X***) for this channel can be computed from:

$$H(Y|X) = \sum_{k=1}^{m}\sum_{j=1}^{n} p(x_k, y_j) \log \frac{1}{p(x_k, y_j)}$$

$$= \sum_{k=1}^{m} p(x_k) \cdot \sum_{j=1}^{n} p(y_j | x_k) \log \frac{1}{p(y_j | x_k)}$$

However, for the channel under consideration observe that:

$$\sum_{k=1}^{m} p(x_k).\sum_{j=1}^{n} p(y_j \mid x_k)\log \frac{1}{p(y_j \mid x_k)} = \sum_{j=1}^{n} p_j \log \frac{1}{p_j} = h \quad \text{..... (4.37)}$$

is a constant, as the entropy function is symmetric with respect to its arguments and depends only on the probabilities but not on their relative locations. Accordingly, the entropy becomes:

$$H(Y \mid X) = \sum_{k=1}^{m} p(x_k).h = h \quad \text{.....(4.38)}$$

as the source probabilities all add up to unity.

Thus the conditional entropy for such type of channels can be computed from the elements of any row of the channel matrix. Accordingly, we have for the mutual information:

$$I(X, Y) = H(Y) - H(Y|X)$$
$$= H(Y) - h$$

Hence, C = Max $I(X, Y)$ =Max $\{H(Y) - h\}$ = Max $H(Y) - h$

Since, $H(Y)$ will be maximum if and only if all the received symbols are equally probable and as there are n – symbols at the output, we have:

$$H(Y)_{Max} = \log n$$

Thus we have for the symmetric channel:

$$C = \log n - h \quad \text{..... (4.39)}$$

The channel matrix of a channel may not have the form described in Eq (3.35) but still it can be a symmetric channel. This will become clear if you interchange the roles of input and output. That is, investigate the conditional probability matrix $P(X|Y)$.

We define the channel to be symmetric if the **CPM**, $P(X|Y)$ has the form:

$$P(X \mid Y) = \begin{matrix} p_1 & p_m & p_2 & \cdots & p_m \\ p_2 & p_1 & p_3 & \cdots & p_{m-1} \\ p_3 & p_4 & p_m & \cdots & p_{m-2} \\ \text{M} & \text{M} & \text{M} & \text{M} & \text{M} \\ p_m & p_1 & p_{m-3} & \cdots & p_1 \end{matrix} \quad \text{.....(4.40)}$$

That is, the second and subsequent columns of the **CPM** are certain permutations of the first column. In other words entries in the second and subsequent columns are exactly the same as in the first column but for different locations. In this case we have:

$$H(X \mid Y) = \sum_{j=1}^{n}\sum_{k=1}^{m} p(x_k, y_j)\log \frac{1}{p(x_k \mid y_j)} = \sum_{j=1}^{n} p(y_j) \sum_{k=1}^{m} p(x_k \mid y_j)\log \frac{1}{p(x_k \mid y_j)}$$

Since $\sum_{j=1}^{n} p(y_j) = 1$ and $\sum_{k=1}^{m} p(x_k \mid y_j)\log \frac{1}{p(x_k \mid y_j)} = \sum_{k=1}^{m} p_k \log \frac{1}{p_k} = h'$ is a constant, because all entries in any column are exactly the same except for their locations, it then follows that:

$$H(X \mid Y) = h' = \sum_{k=1}^{m} p_k \log \frac{1}{p_k} \quad \text{..... (4.41)}$$

*Remember that the sum of all entries in any column of Eq (3.40) should be unity.

As a consequence, for the symmetry described we have:

$$C = \text{Max } [H(X) - H(X|Y)] = \text{Max } H(X) - h'$$

Or $$C = \log m - h' \quad \text{.....(4.42)}$$

Thus the channel capacity for a symmetric channel may be computed in a very simple and straightforward manner. Usually the channel will be specified by its noise characteristics and the source probabilities [i.e. $P(Y|X)$ and $P(X)$]. Hence it will be a matter of simple inspection to identify the first form of symmetry described. To identify the second form of symmetry you have to first compute $P(X|Y)$ – tedious!

Example 4.4:

Consider the channel represented by the channel diagram shown in Fig 3.3:

The channel matrix can be read off from the channel diagram as:

$$P(Y \mid X) = \begin{matrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} \end{matrix}$$

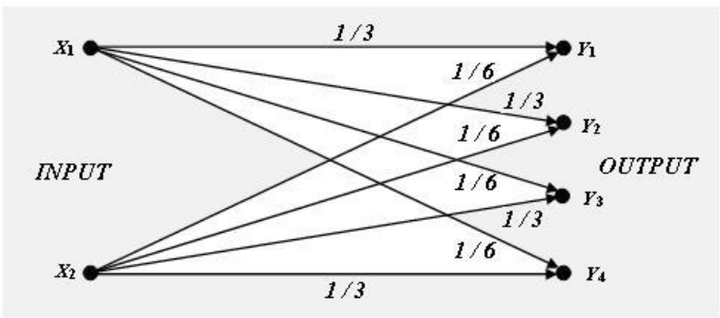


Fig 4.3 A Symmetric Channel

Clearly, the second row is a permutation of the first row (written in the reverse order) and hence the channel given is a symmetric channel. Accordingly we have, for the noise entropy, h (from either of the rows):

$$H(Y|X) = h = 2 \times \frac{1}{3} \log 3 + 2 \times \frac{1}{6} \log 6 = 1.918295834 \text{ bits / symbol.}$$

$$C = \log n - h = \log 4 - h = 0.081704166 \text{ bits / symbol.}$$

Example 4.5:

A binary channel has the following noise characteristic:

$$\begin{array}{c|cc} & 0 & 1 \\ 0 & \frac{2}{3} & \frac{1}{3} \\ 1 & \frac{1}{3} & \frac{2}{3} \end{array}$$

(a) If the input symbols are transmitted with probabilities $3/4$ and $1/4$ respectively, find $H(X)$, $H(Y)$, $H(X, Y)$, $H(Y|X)$ and $I(X, Y)$.
(b) Find the channel capacity, efficiency and redundancy of the channel.
(c) What are the source probabilities that correspond to the channel capacity?

To avoid confusion, let us identify the input symbols as x_1 and x_2 and the output symbols by y_1 and y_2. Then we have:

$P(x_1) = 3/4$ and $p(x_2) = 1/4$

$$P(X|Y) = \begin{matrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{matrix}$$

$$H(Y|X) = h = \frac{2}{3}\log\frac{3}{2} + \frac{1}{3}\log 3 = \log 3 - \frac{2}{3} = 0.918295833 \text{ bits / symbol}.$$

$$H(X) = \frac{3}{4}\log\frac{4}{3} + \frac{1}{4}\log 4 = \log 4 - \frac{3}{4}\log 3 = 2 - \frac{3}{4}\log 3 = 0.811278125 \text{ bits / symbol}.$$

Multiplying first row of $P(Y|X)$ by $p(x_1)$ and second row by $p(x_2)$ we get:

$$P(X,Y) = \begin{matrix} \frac{2}{3}\times\frac{3}{4} & \frac{1}{3}\times\frac{3}{4} \\ \frac{1}{3}\times\frac{1}{4} & \frac{2}{3}\times\frac{3}{4} \end{matrix} = \begin{matrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{12} & \frac{1}{6} \end{matrix}$$

Adding the elements of this matrix columnwise, we get: $p(y_1) = 7/12$, $p(y_2) = 5/12$.

Dividing the first column entries of $P(X, Y)$ by $p(y_1)$ and those of second column by $p(y_2)$, we get:

$$P(X|Y) = \begin{matrix} \frac{6}{7} & \frac{3}{5} \\ \frac{1}{7} & \frac{2}{5} \end{matrix}$$

From these values we have:

$$H(Y) = \frac{7}{12}\log\frac{12}{7} + \frac{5}{12}\log\frac{12}{5} = 0.979868756 \text{ bits / symbol}.$$

$$H(X,Y) = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{12}\log 12 + \frac{1}{6}\log 6 = 1.729573958 \text{ bits / symbol}.$$

$$H(X|Y) = \frac{1}{2}\log\frac{7}{6} + \frac{1}{4}\log\frac{5}{3} + \frac{1}{12}\log 7 + \frac{1}{6}\log\frac{5}{2} = 0.74970520 \text{ bits / symbol}$$

$$H(Y|X) = \frac{1}{2}\log\frac{3}{2} + \frac{1}{4}\log 3 + \frac{1}{12}\log 3 + \frac{1}{6}\log\frac{3}{2} = \log 3 - \frac{2}{3} = h \quad (\text{as before}).$$

$$I(X,Y) = H(X) - H(X|Y) = 0.061572924 \text{ bits/symbol}.$$

$$= H(Y) - h = 0.061572923 \text{ bits/symbol}.$$

$$C = \log n - h = \log 2 - h - 1 - h = 0.081704167 \text{ bits/symbol}.$$

$$\text{Efficiency}, \eta = \frac{I(X,Y)}{C} = 0.753608123 \text{ or } 75.3608123\%$$

$$\text{Redundancy}, E = 1 - \eta = 0.246391876 \text{ or } 24.6391876\%$$

To find the source probabilities, let $p(x_1) = p$ and $p(x_2) = q = 1 - p$ .Then the ***JPM*** becomes:

$$P(X,Y) = \begin{matrix} \frac{2}{3}p, & \frac{1}{3}p \\ \frac{1}{3}(1-p), & \frac{2}{3} \end{matrix}$$

Adding columnwise we get: $p(y_1) = \frac{1}{3}(1+p)$ *and* $p(y2) = \frac{1}{3}(2-p)$

For $H(Y) = H(Y)_{\max}$, we want $p(y_1) = p(y_2)$ and hence $1+p = 2-p$ or $p = \frac{1}{2}$

Therefore the source probabilities corresponding to the channel capacity are: $p(x_1) = 1/2 = p(x_2)$.

Binary Symmetric Channels (BSC): (Problem 2.6.2 – S imon Haykin)

The channel considered in Example 3.6 is called a 'Binary Symmetric Channel' or (***BSC***). It is one of the most common and widely used channels. The channel diagram of a ***BSC*** is shown in Fig 3.4. Here ' ***p***' is called the error probability.

For this channel we have:

$$H(Y|X) = p\log\frac{1}{p} + q\log\frac{1}{q} = H(p) \qquad (4.43)$$

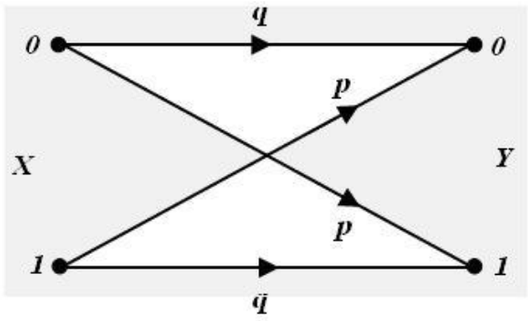
$$H(Y) = [p - \alpha(p-q)]\log\frac{1}{[p-\alpha(p-q)]} + [q + \alpha(p-q)]\log\frac{1}{[q+\alpha(p-q)]} \quad \ldots(4.44)$$

$I(X, Y) = H(Y) - H(Y|X)$ and the channel capacity is:

$$C = 1 + p\log p + q\log q \qquad \ldots\ldots\ldots\ldots(4.45)$$

This occurs when $\alpha = 0.5$ *i.e.* $P(X=0) = P(X=1) = 0.5$

In this case it is interesting to note that the equivocation, $H(X|Y) = H(Y|X)$.



$$P(Y|X) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} q & p \\ q & p \end{bmatrix} \end{array}$$

$p + q = 1$
$P(X=0) = \alpha$
$P(X=1) = 1-\alpha$

Fig 4.4 Channel diagram of a BSC and its Channel Matrix.

An interesting interpretation of the equivocation may be given if consider an idealized communication system with the above symmetric channel as shown in Fig 4.5.

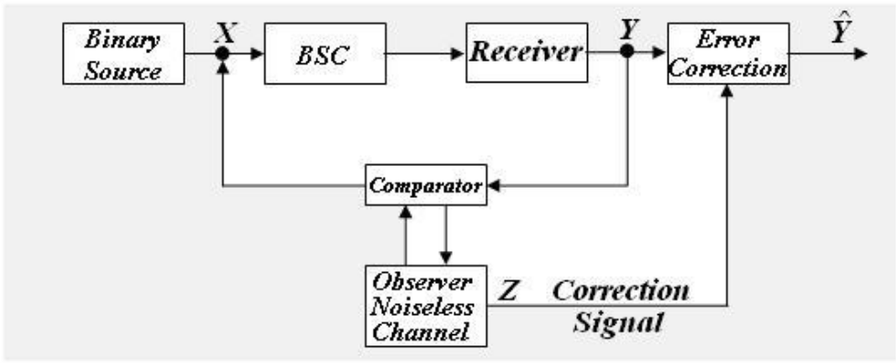


Fig 4.5 An Idealized Binary Communication System.

The observer is a noiseless channel that compares the transmitted and the received symbols. Whenever there is an error a '*1*' is sent to the receiver as a correction signal and appropriate correction is effected. When there is no error the observer transmits a '*0*' indicating no change. Thus the observer supplies additional information to the receiver, thus compensating for the noise in the channel. Let us compute this additional information .With $P(X=0) = P(X=1) = 0.5$, we have:

Probability of sending a '1' = Probability of error in the channel .

$$\text{Probability of error} = P(Y=1|X=0).P(X=0) + P(Y=0|X=1).P(X=1)$$
$$= p \times 0.5 + p \times 0.5 = p$$
$$\therefore \text{Probability of no error} = 1 - p = q$$

Thus we have P $(Z = 1) = p$ and $P(Z = 0) = q$

Accordingly, additional amount of information supplied is:

$$= p \log \frac{1}{p} + q \log \frac{1}{q} = H(X|Y) = H(Y|X) \qquad \text{..... (4.46)}$$

Thus the additional information supplied by the observer is exactly equal to the equivocation of the source. Observe that if '*p*' and '*q*' are interchanged in the channel matrix, the trans -information of the channel remains unaltered. The variation of the mutual information with the probability of error is

shown in Fig 3.6(*a*) for ***P (X=0) = P (X=1) = 0.5***. In Fig 4.6(*b*) is shown the dependence of the mutual information on the source probabilities.

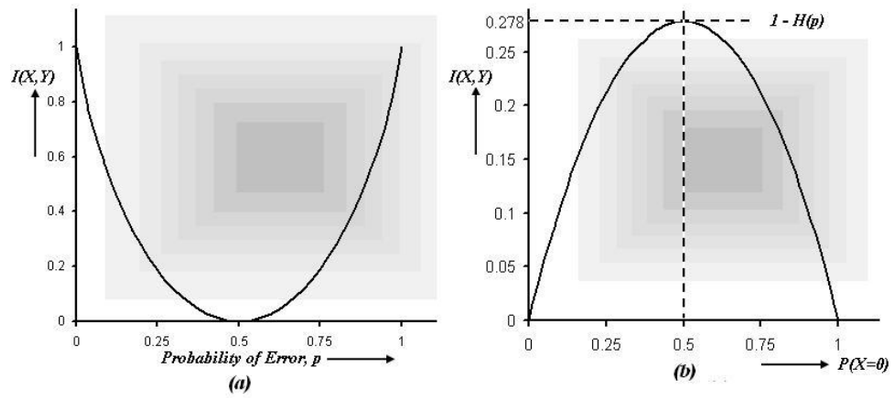
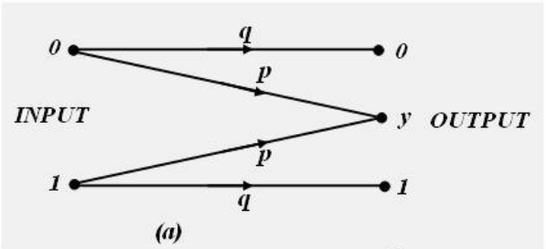


Fig 4.6 *Mutual Information of a BSC*

4.4.4 Binary Erasure Channels (*BEC*):(Problem 2.6.4 – Simon Haykin)

The channel diagram and the channel matrix of a ***BEC*** are shown in Fig 3.7.



$$P(Y|X) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & y & 1 \end{array} \\ \begin{bmatrix} q & p & 0 \\ 0 & p & q \end{bmatrix} \end{array}$$

(b)

Fig 4.7 *Binary Erasure Channel.*

BEC is one of the important types of channels used in digital communications. Observe that whenever an error occurs, the symbol will be received as ' ***y***' and no decision will be made about the information but an immediate request will be made for retransmission, rejecting what have been received (***ARQ*** techniques), thus ensuring ***100%*** correct data recovery. Notice that this channel also is a symmetric channel and we have with $P(X = 0) = \alpha$, $P(X = 1) = 1 - \alpha$.

$$H(Y|X) = p\log\frac{1}{p} + q\log\frac{1}{q} \quad \ldots\ldots\ldots\ldots \quad (4.47)$$

$$H(X) = \alpha\log\frac{1}{\alpha} + (1-\alpha)\log\frac{1}{(1-\alpha)} \quad \ldots\ldots\ldots\ldots \quad (4.48)$$

The ***JPM*** is obtained by multiplying first row of ***P (Y|X)*** by α and second row by $(1-\alpha)$.
We get:

$$P(X,Y) = \begin{matrix} q\alpha & p\alpha & 0 \\ 0 & p(1-\alpha) & q(1-\alpha) \end{matrix} \quad \ldots\ldots\ldots\ldots \quad (4.49)$$

Adding column wise we get: $P(Y) = [q\alpha, p, q(1-\alpha)]$ ..... (4.50)
From which the CPM ***P (X|Y)*** is computed as:

$$P(X|Y) = \begin{matrix} 1 & \alpha & 0 \\ 0 & (1-\alpha) & 1 \end{matrix} \quad \ldots\ldots\ldots\ldots\ldots \quad (4.51)$$

$$\therefore H(X|Y) = \alpha q \log 1 + \alpha p \log \frac{1}{\alpha} + (1-\alpha) p \log \frac{1}{(1-\alpha)} + (1-\alpha) q \log 1$$

$$= pH(X)$$

$$\therefore I(X, Y) = H(X) - H(X|Y) = (1-p)\, H(X) = q\, H(X) \quad \ldots\ldots\ldots \quad (4.52)$$

$$\therefore C = Max\, I(X, Y) = q \text{ bits / symbol.} \quad \ldots\ldots\ldots\ldots \quad (4.53)$$

In this particular case, use of the equation $I(X, Y) = H(Y) - H(Y|X)$ will not be correct, as $H(Y)$ involves 'y' and the information given by 'y' is rejected at the receiver.

Deterministic and Noiseless Channels: (Additional Information)

Suppose in the channel matrix of Eq (3.34) we make the following modifications.

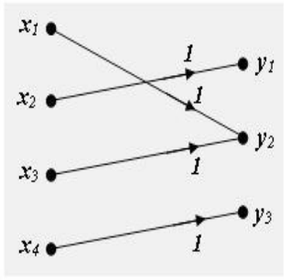
a) Each row of the channel matrix contains one and only one nonzero entry, which necessarily should be a '*1*'. That is, the channel matrix is symmetric and has the property, for a given k and j, $P(y_j|x_k) = 1$ and all other entries are '**0**'. Hence given x_k, probability of receiving it as y_j is ***one***. For such a channel, clearly

$$H(Y|X) = 0 \text{ and } I(X, Y) = H(Y) \quad \ldots\ldots\ldots\ldots\ldots \quad (4.54)$$

Notice that it is not necessary that $H(X) = H(Y)$ in this case. The channel with such a property will be called a '***Deterministic Channel***'.

Example 4.6:

Consider the channel depicted in Fig 3.8. Observe from the channel diagram shown that the input symbol x_k uniquely specifies the output symbol y_j with a probability one. By observing the output, no decisions can be made regarding the transmitted symbol!!



$$P(Y|X) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fig 4.8

b) Each column of the channel matrix contains one and only one nonzero entry. In this case, since each column has only one entry, it immediately follows that the matrix $P(X|Y)$ has also one and only one non zero entry in each of its columns and this entry, necessarily be a '*1*' because:

If $p(y_j|x_k) = \alpha$, $p(y_j | x_r) = 0$, $r \neq k$, $r = 1, 2, 3 \ldots m$.

Then $p(x_k, y_j) = p(x_k) \times p(y_j|x_k) = \alpha \times p(x_k)$,

$$p(x_r, y_j) = 0,\ r \neq k,\ r = 1, 2, 3 \ldots \quad m.$$

$$\therefore p(y_j) = \sum_{r=1}^{m} p(x_r, y_j) = p(x_k, y_j) = \alpha\, p(x_k)$$

$$\therefore p(x_k \mid y_j) = \frac{p(x_k, y_j)}{p(y_j)} = 1, \text{ and } p(x_r \mid y_j) = 0, \quad \forall r \neq k, r = 1,2,3,\ldots m.$$

It then follows that $H(X|Y) = 0$ and $I(X, Y) = H(X)$ ..... (4.55)

Notice again that it is not necessary to have $H(Y) = H(X)$. However in this case, converse of (*a*) holds. That is one output symbol uniquely specifies the transmitted symbol, whereas for a given input symbol we cannot make any decisions about the received symbol. The situation is exactly the complement or mirror image of (*a*) and we call this channel also a deterministic channel (some people call the channel pertaining to case (*b*) as 'Noiseless Channel', a classification can be found in the next paragraph). Notice that for the case (*b*), the channel is symmetric with respect to the matrix $P(X|Y)$.

Example 4.7:

Consider the channel diagram, the associated channel matrix, $P(Y|X)$ and the conditional probability matrix $P(X|Y)$ shown in Fig 3.9. For this channel, let

$p(x_1)=0.5, p(x_2) = p(x_3) = 0.25.$

Then $p(y_1) = p(y_2) = p(y_6) = 0.25, p(y_3) = p(y_4) = 0.0625$ and $p(y_5) = 0.125.$

It then follows $I(X, Y) = H(X) = 1.5$ *bits / symbol*,

$H(Y) = 2.375$ *bits / symbol*, $H(Y|X) = 0.875$ *bits / symbol* and $H(X|Y) = 0.$

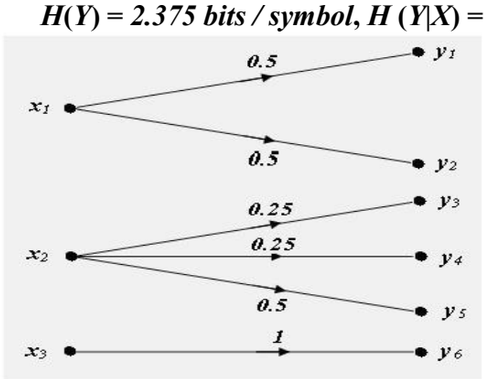


Fig 4.9 Channel diagram for Example 4.7

$$P(Y|X) = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0.25 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore P(X|Y) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(Irrespective of P(X), the reader should verify this)

c) ***Now let us consider a special case***: The channel matrix in Eq (3.34) is a square matrix and all entries except the one on the principal diagonal are zero. That is:

$$p(y_k|x_k) = 1 \text{ and } p(y_j|x_k) = 0\ \forall k \neq j$$

Or in general, $p(y_j|x_k) = \delta_{jk}$, where δ_{jk}, is the '***Kronecker delta***', i.e. $\delta_{jk} = 1$ *if* $j = k$
$= 0$ *if* $j \neq k.$

That is, $P(Y|X)$ is an Identity matrix of order 'n' and that $P(X|Y) = P(Y|X)$ and $p(x_k, y_j) = p(x_k) = p(y_j)$ can be easily verified.

For such a channel it follows:

$H(X|Y) = H(Y|X) = 0$ and $I(X, Y) = H(X) = H(Y) = H(X, Y)$ .....… (4.56)

We call such a channel as " ***Noiseless Channel***". Notice that for the channel to be noiseless, it is necessary that there shall be a one-one correspondence between input and output symbols. No information will be lost in such channels and if all the symbols occur with equal probabilities, it follows then:

$$C = I(X, Y)_{Max} = H(X)_{Max} = H(Y)_{Max} = \log n \text{ bits / symbol.}$$

Thus a noiseless channel is symmetric and deterministic with respect to both descriptions $P(Y|X)$ and $P(X|Y)$.

Finally, observe the major concept in our classification. In case (***a***) for a given transmitted symbol, we can make a unique decision about the received symbol from the source end. In case (***b***), for a given received symbol, we can make a decision about the transmitted symbol from the receiver end. Whereas for case (***c***), a unique decision can be made with regard to the transmitted as well as the received symbols from either ends. This uniqueness property is vital in calling the channel as a 'Noiseless Channel'.

d) To conclude, we shall consider yet another channel described by the following ***JPM***:

$$P(X,Y) = \begin{matrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{matrix} \begin{matrix} p_1 & p_1 & \cdots & p_1 \\ p_2 & p_2 & \cdots & p_2 \\ \vdots & \vdots & \cdots & \vdots \\ p_m & p_m & \cdots & p_m \end{matrix}$$

$$\text{with } \sum_{k=1}^{m} p_k = \frac{1}{n} \quad i.e.\ p(y_j) = \frac{1}{n},\ \forall\, j = 1,2,3,\ldots n.$$

This means that there is no correlation between x_k and y_j and an input x_k may be received as any one of the y_j's with equal probability. In other words, the input-output statistics are independent!!

This can be verified, as we have $p(x_k, y_j) = p_k$

$$= np_k \cdot \sum_{k=1}^{m} p_k = p(x_k).p(y_j)$$

$\therefore\ p(x_k|y_j) = np_k$ and $p(y_j|x_k) = 1/n$

Thus we have:

$$H(X,Y) = n.\sum_{k=1}^{m} p_k \log,\ H(X) = \sum_{k=1}^{m} np_k \log \frac{1}{np_k} = -n \sum_{k=1}^{m} p_k \log \frac{1}{p_k} + \log \frac{1}{n}$$

$$H(Y) = \sum_{j=1}^{n} p(y_j) \log \frac{1}{p(y_j)} = \log n,$$

$$H(X|Y) = H(X),\ H(Y|X) = H(Y) \text{ and } I(X,Y) = 0$$

$\therefore$ Such a channel conveys no information whatsoever. Thus a channel with independent input-output structure is similar to a network with largest internal loss (purely resistive network), in contrast to a noiseless channel which resembles a lossless network.

Some observations:

For a deterministic channel the noise characteristics contains only one nonzero entry, which is a ' ***1***', in each row or only one nonzero entry in each of its columns. In either case there exists a linear dependence of either the rows or the columns. For a noiseless channel the rows as well as the columns of the noise characteristics are linearly independent and further there is only one nonzero entry in each row as well as each column, which is a ' ***1***' that appears only on the principal diagonal (or it may be on the skew diagonal). For a channel with independent input-output structure, each row and column are made up of all nonzero entries, which are all equal and equal to ***1/n***. Consequently both the rows and the columns are always linearly dependent!!

Franklin.M.Ingels makes the following observations:

1) If the channel matrix has only one nonzero entry in each column then the channel is termed as " ***loss-less channel***". True, because in this case $H(X|Y) = 0$ and $I(X, Y) = H(X)$, i.e. the mutual information equals the source entropy.

2) If the channel matrix has only one nonzero entry in each row (which necessarily should be a ' ***1'***), then the channel is called " ***deterministic channel***". In this case there is no ambiguity about how the transmitted symbol is going to be received although no decision can be made from the receiver end. In this case $H(Y|X) = 0$, and $I(X, Y) = H(Y)$.

3) An " ***Ideal channel***" is one whose channel matrix has only one nonzero element in each row and each column, i.e. a diagonal matrix. An ideal channel is obviously both loss-less and deterministic. Lay man's knowledge requires equal number of inputs and outputs-you cannot transmit 25 symbols and receive either 30 symbols or 20 symbols, there shall be no difference between the numbers of transmitted and received symbols. In this case

   $I(X,Y) = H(X) = H(Y)$; and $H(X|Y) = H(Y|X) = 0$

4) A " ***uniform channel***" is one whose channel matrix has identical rows ex cept for permutations OR identical columns except for permutations. If the channel matrix is square, then every row and every column are simply permutations of the first row.

Observe that it is possible to use the concepts of " *sufficient reductions*" and make the channel described in (1) a deterministic one. For the case (4) observe that the rows and columns of the matrix (Irreducible) are linearly independent.

Additional Illustrations:

Example 4.8:

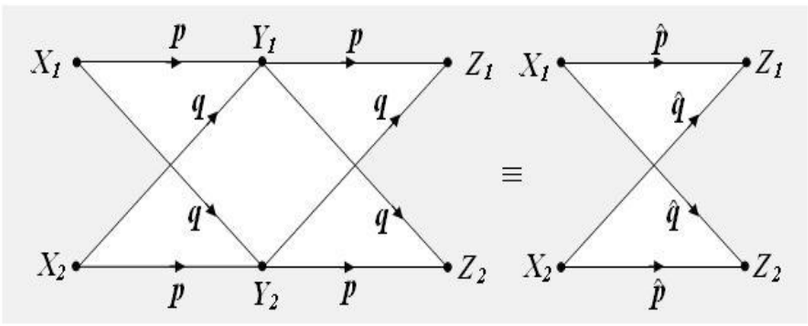


Fig 4.10 *Two BSC's in Cascade*

Consider two identical ***BSC's*** cascaded as shown in Fig 4.10. Tracing along the transitions indicated we find:

$$p(z_1|x_1) = p^2 + q^2 = (p+q)^2 - 2pq = (1-2pq) = p(z_2|x_2) \text{ and } p(z_1|x_2) = 2pq = p(z_2|x_1)$$

Labeling $\hat{p} = 1 - 2pq$, $\hat{q} = 2pq$ it then follows that:

$$I(X, Y) = 1 - H(q) = 1 + p \log p + q \log q$$
$$I(X, Z) = 1 - H(2pq) = 1 + 2pq \log 2pq + (1 - 2pq) \log (1 - 2pq).$$

If one more identical ***BSC*** is cascaded giving the output (u_1, u_2) we have:

$$I(X, U) = 1 - H(3pq^2 + p^3)$$

The reader can easily verify that $I(X, Y) \geq I(X, Z) \geq I(X, U)$

Example 4.9:

Let us consider the cascade of two noisy channels with channel matrices:

$$P(Y|X) = \begin{matrix} \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{matrix} \qquad P(Z|Y) = \begin{matrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} \end{matrix}$$, with $p(x_1) = p(x_2) = 0.5$

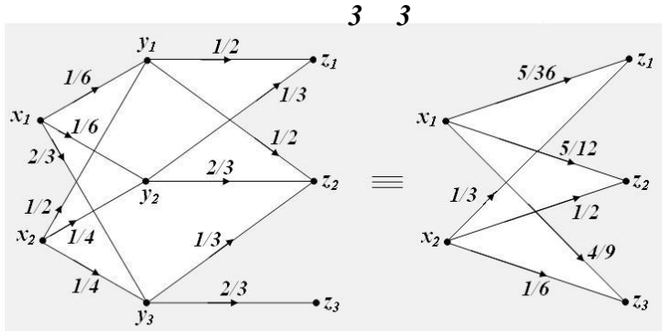


Fig 4.11 *Cascade of two noisy channels.*

The above cascade can be seen to be equivalent to a single channel with channel matrix:

$$P(Z|X) = \begin{matrix} \frac{5}{36} & \frac{5}{12} & \frac{4}{9} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{6} \end{matrix}$$

The reader can verify that: ***I(X, Y) = 0.139840072 bits / symbol.***

I(X, Z) = 0.079744508 bits / symbol.

Clearly ***I(X, Y) > I(X, Z).***

Example 4.10: Let us consider yet another cascade of noisy channels described by:

$$P(Y|X) = \begin{matrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{matrix} \qquad P(Z|Y) = \begin{matrix} 1 & 0 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \end{matrix}$$

The channel diagram for this cascade is shown in Fig 4.12. The reader can easily verify in this case that the cascade is equivalent to a channel described by:

$$P(Z|X) = \begin{matrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{matrix} = P(Y|X);$$

Inspite of the fact, that neither channel is noiseless, here we have ***I(X, Y) = I(X, Z).***

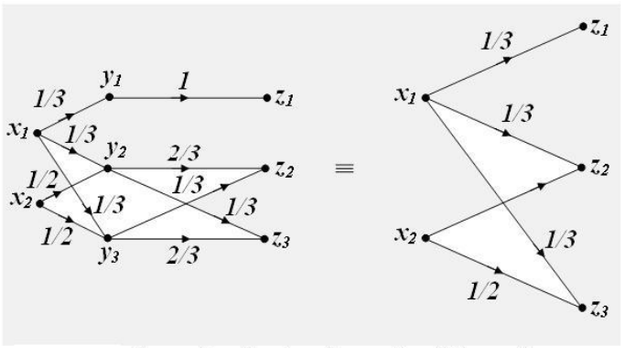


Fig 4.12 *Cascade of noisy channels of Example* **4.10**

Review Questions:

1. What are important properties of the codes?

2. what are the disadvantages of variable length coding?

3. Explain with examples:
4. Uniquely decodable codes, Instantaneous codes

5. Explain the Shannon-Fano coding procedure for the construction of an optimum code

6. Explain clearly the procedure for the construction of compact Huffman code.

7. A discrete source transmits six messages symbols with probabilities of 0.3, 0.2, 0.2, 0.15, 0.1, 0.5. Device suitable Fano and Huffmann codes for the messages and determine the average length and efficiency of each code.

8. Consider the messages given by the probabilities 1/16, 1/16, 1/8, ¼, ½. Calculate H. Use the Shannon-Fano algorithm to develop a efficient code and for that code, calculate the average number of bits/message compared with H.

9. Consider a source with 8 alphabets and respective probabilities as shown:
   A B C D E F G H
   0.20 0.18 0.15 0.10 0.08 0.05 0.02 0.01
   Construct the binary Huffman code for this. Construct the quaternary Huffman and code and show that the efficiency of this code is worse than that of binary code

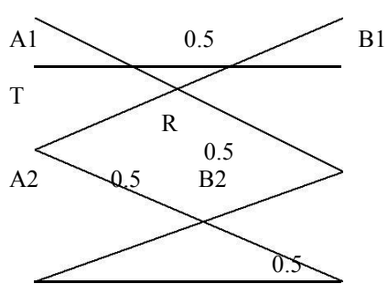
10. Define Noiseless channel and deterministic channel.

11. A source produces symbols X, Y,Z with equal probabilities at a rate of 100/sec. Owing to noise on the channel, the probabilities of correct reception of the various symbols are as shown:

P (j/i)	X	Y	z
X	¾	¼	0
y	¼	½	¼
z	0	¼	¾

Determine the rate at which information is being received.

12. Determine the rate of transmission I(x,y) through a channel whose noise characteristics is shown in fig. P(A1)=0.6, P(A2)=0.3, P(A3)=0.1



Unit – 4

Syllabus:

Channel coding theorem, Differential entropy and mutual information for continuous ensembles, Channel capacity Theorem. **6 Hours**

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996.
Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007
Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

Unit – 4

CONTINUOUS CHANNELS

Until now we have considered discrete sources and discrete channels that pertain to a digital communication system (pulse code modulation). Although modern trend is to switch over to digital communications, analog communications can never become obsolete where amplitude and frequency modulations are used (Radio, television etc). Here the modulating signal ***X (t)*** (which is the set of messages to be transmitted from an information theoretical view point) is invariably a continuous speech or picture signal. This message can be treated as equivalent to a continuous sample space whose sample points form a continuum, in contrast to the discrete case. We shall define a continuous channel as one whose input is a sample point from a continuous sample space and the output is a sample point belonging to either the same sample space or to a different sample space. Further we shall define a' ***zero memory continuous channel***' as the one in which the channel output statistically depends on the corresponding channels without memory. In what follows, we briefly discuss the definition of information and entropy for the continuous source, omitting the time dependence of the messages for brevity and conclude with a discussion of Shannon-Hartley law.

Entropy of continuous Signals: (Differential entropy):

For the case of discrete messages, we have defined the entropy as

$$H(S) = \sum_{k=1}^{q} p(s_k) \log \frac{1}{p(s_k)} \qquad \text{..... (5.1)}$$

For the case of analog or continuous messages, we may wish to extend Eq. (5.1), considering the analog data to be made up of infinite number of discrete messages. However, as $P(X = x) = 0$ for a ***CRV, X***, then direct extension of Eq. (5.1) may lead to a meaningless definition. We shall proceed as follows:

Let us consider that the continuous source ' ***X***'(a continuous random variable) as a limiting form of a discrete random variable that assumes discrete values $0, \pm\Delta x, \pm 2\Delta x, \ldots$,etc. Let $k.\Delta x = x_k$, then clearly $\Delta x_k = \Delta x$. The random variable **X** assumes a value in the range $(x_k, x_k+\Delta x)$ with probability $f(x_k) \times \Delta x_k$, (recall that $P\{x \le X \le x+dx\} = f(x).dx$, the alternative definition of the p.d.f. of an r.v). In the limit as $\Delta x_k \to 0$, the error in the approximation would tend to become zero. Accordingly, the entropy of the ***CRV, X***, is given by, using Eq. (5.1),

$$H(X) = \lim_{\Delta x_k \to 0} \sum_{\forall k} f(x_k)\Delta x_k \log \frac{1}{f(x_k)\Delta x_k}$$

$$= \lim_{\Delta x_k \to 0} \sum_{\forall k} f(x_k)\Delta x_k \log \frac{1}{f(x_k)} - \sum_{\forall k} f(x_k)\Delta x_k \log \Delta x_k$$

As $\Delta x_k \to 0$ it follows: $x_k \to x$ and $f(x_k) \times \Delta x_k \to f(x)\,dx$. The summation would be replaced by ***integration***, and thus we have:

$$H(X) = \int_{-\infty}^{\infty} f(x) \log \frac{1}{f(x)} dx - \lim_{\Delta x_k \to 0} \log \Delta x \int_{k}^{\infty} f(x)dx$$

$$= \int_{-\infty}^{\infty} f(x) \log \frac{1}{f(x)} dx - \lim_{\Delta x \to 0} \log \Delta x \quad \text{..... (5.2)}$$

As $\Delta x_k = \Delta x$, and the second integral in the RHS is unity (property of any density function). Notice that in the limit as $\Delta x \to 0$, $\log \Delta x \to$ '$-\infty$' . Thus, it appears that the entropy of a continuous random variable is infinite. Indeed it is so. The amount of uncertainty associated with a continuous message is infinite – as a continuous signal assumes uncountab ly infinite values. Thus it seems there can be no meaningful definition of entropy for a continuous source (notice that we are using the words source, message and random variable interchangeably as we are referring to statistical data that is being generated by the transmitter). However, the anomaly can be solved if we consider the first term in the RHS of Eq. (5.2) as a relative measure while '$-\log \Delta x$' serves as our reference. Since we will be dealing, in general, with differences in entropies (for example $I(X, Y) = H(X) - H(X|Y)$), if we select the same datum for all entropies concerned, the relative measure would be indeed quite meaningful. However, caution must be exercised to remember that it is only a relative measure and not an absolute value. Otherwise, this subtle point would lead to many apparent fallacies as can be observed by the following example. In order to differentiate from the ordinary absolute entropy we call it ' ***Differential entropy***'. We then define the entropy. ***H(X)***, of a continuous source as

$$H(X) \triangleq \int_{-\infty}^{\infty} f(x) \log \frac{1}{f(x)} dx \quad \text{bits / sample .} \quad \text{..... (5.3)}$$

Where ***f(x)*** is the probability density function (***p.d.f***.) of the ***CRV, X***.

Example 5.1: Suppose ***X*** is a uniform r.v. over the interval (***0, 2***). Hence

$$f(x) = \frac{1}{2} \qquad 0 \le x \le 2$$

$$= 0 \quad \text{Else where}$$

Then using Eq. (5.3), we have $H(X) = \int_0^2 \frac{1}{2} \log 2\, dx = 1 \text{bit / sample}$

Suppose ***X*** is the input to a linear amplifier whose gain = ***4***. Then the output of the amplifier would be ***Y = 4X***. Then it follows ***f(y) = 1/8, 0 ≤ y ≤ 8***. Therefore the entropy ***H(Y)***, of the amplifier output

becomes $H(Y) = \int_0^8 \frac{1}{8} \log 8\, dy = 3 \text{bits / sample}$

That is the entropy of the output is thrice that of the input! However, since knowledge of ***X*** uniquely determines ***Y***, the average uncertainties of ***X*** and ***Y*** must be identical. Definitely, amplification of a signal can neither add nor subtract information. This anomaly came into picture because we did not bother about the reference level. The reference entropies of ***X*** and ***Y*** are:

$$R_x = \lim_{\Delta x \to 0} (-\log \Delta x) \quad \text{and} \quad R_y = \lim_{\Delta y \to 0} (-\log \Delta y)$$

$$R_x - R_y = \lim_{\substack{\Delta x \to 0 \\ \Delta y \to 0}} (-\log \Delta x + \log \Delta y) = \log \frac{dy}{dx} = \log 4 = 2 \text{bits/sample}.$$

Clearly, the reference entropy of X, R_x is higher than that of Y, R_y. Accordingly, if X and Y have equal absolute entropies, then their relative entropies must differ by 2 **bits**. We have:

Absolute entropy of $X = R_x + H(X)$
Absolute entropy of $Y = R_y + H(Y)$

Since $R_x = R_y + 2$, the two absolute entropies are indeed equal. This conclusion is true for any reverse operation also. However, the relative entropies will be, in general, different.

To illustrate the anomalies that one may encounter if the relative characterization of the entropy function is ignored we shall consider another example.

Example 5.2:

Suppose X is uniformly distributed over the interval $(-1/4, 1/4)$ .Then

$f(x) = 2$ …. $-1/4 \le x \le 1/4$ and ‘ 0’ else where. It then follows from Eq (5.3):

$$H(X) = \int_{-1/4}^{1/4} 2 \log \frac{1}{2} dx = -\log 2 = -1 \text{bit/sample}.$$

which is ‘negative’ and very much contradictory to our concept of information?

Maximization of entropy:

For the case of discrete messages we have seen that entropy would become a maximum when all the messages are equally probable. In practical systems, the sources, for example, radio transmitters, are constrained to either average power or peak power limitations. Our objective, then, is to maximize the entropy under such restrictions. The general constraints may be listed as below:

1) Normalizing constraint: $\int_{-\infty}^{\infty} f(x)dx = 1$ (Basic property of any density function)

2) Peak value limitation: $\int_{-M}^{M} f(x)dx = 1$

3) Average value limitation $\int_{-\infty}^{\infty} x.f(x)dx = \mu = E\{X\}$, is a constant.

4) Average power limitation $\int_{-\infty}^{\infty} x^2.f(x)dx = m_2 = E\{X^2\}$, is a constant.

5) Average power limitation, with unidirectional distribution (causal systems whose response does not begin before the input is applied) $\int_0^\infty x^2 . f(x)dx = \alpha$, a constant.

Our interest then is to find the *p.d.f.*, ***f(x)*** that maximizes the entropy function ***H(X)*** defined in Eq. (5.3). The maximum entropy method (***MEM***) we are speaking of then is a problem of constrained optimization and is a particular case of the so-called isoperimetric problem of the calculus of variations. We shall adopt, here, the 'Euler- Lagrange's' method of undetermined co-efficients to solve the problem on hand. Suppose, we wish to maximize the integral

$$I = \int_a^b \varphi(x . f)dx \qquad \text{..... ..... (5.4)}$$

Subject to the following integral constraints:

$$\int_a^b \varphi_1 (x , f)dx = \lambda_1$$

$$\int_a^b \varphi_2 (x , f)dx = \lambda_2 \qquad \text{..... ..... (5.5)}$$

M
M

$$\int_a^b \varphi_r \quad (x , f)dx = \lambda_r$$

Where $\lambda_1, \lambda_2 \ldots \lambda_r$ are pre-assigned constants. Then the form of ***f(x)*** that satisfies all the above constraints and makes ' ***I***' a maximum (or minimum) is computed by solving the equation

$$\frac{\partial \varphi}{\partial f} + \alpha_1 \frac{\partial \varphi_1}{\partial f} + \alpha_2 \frac{\partial \varphi_2}{\partial f} + \ldots + \alpha_r \frac{\partial \varphi_r}{\partial f} = 0 \qquad \text{.....(5.6)}$$

The undetermined co-efficients, $\alpha_1, \alpha_2 \ldots \alpha_r$ are called the 'Lagrangian multipliers' (or simply, Lagrangians) are determined by substituting the value of ***f(x)*** in Eq (5.5) successively. (Interested reader can refer standard books on ' *calculus of variations*' for such optimization problems).We shall consider the cases we have already listed and determine the *p.d.f*, ***f(x)*** of the random signal ***X*** for these cases.

Since differentiation and integration of ***log f(x)*** is not possible directly, we convert it into logarithm to base ' ***e***' and proceed as below:

$\log_2 f = \log e . \ln f = \ln f^a = \ln f'$, where $a = \log e$, a constant and eventually we will be finding f' that maximizes Eq (5.7) or equivalently ***f***, if we call $H_1(x) = -\int f(x) \ln f(x)\, dx$, then $H(X) = (\log e)\, H_1(X)$. So if we maximize $H_1(X)$, we will have achieved our goal.

Case I: Peak Signal Limitation:

Suppose that the signal is peak limited to ±***M*** (Equivalent to peak power limitation, S_m, as in the case of ***AM***, ***FM*** and Pulse modulation transmitters. For example in ***FM***, we use limiting circuits as our concern is only on frequency deviation.). Then we have,

$$H_1(X) = -\int_{-M}^{M} f(x) \log f(x)\,dx \qquad \text{..... (5.7)}$$

And the only constraint on ***f(x)*** being the unit area condition (property of a ***p.d.f***)

$$\int_{-M}^{M} f(x)\,dx = 1 \qquad \text{..... (5.8)}$$

Here $\varphi(x, f) = -f \ln f \quad \Rightarrow \quad \frac{\partial \varphi}{\partial f} = -(1 + \ln f)$

$$\varphi_1(x, f) = f \quad \Rightarrow \quad \frac{\partial \varphi}{\partial f_1} = 1$$

And from Eq. (5.6), we have

$$\frac{\partial \varphi}{\partial f} + \alpha_1 \frac{\partial \varphi}{\partial f_1} = 0 \qquad \text{..... (5.9)}$$

$$\Rightarrow -(1+\ln f) + \alpha = 0 \Rightarrow f = e^{-(1-\alpha)} \qquad \text{..... (5.10)}$$

Substituting Eq (5.10) in Eq (5.8), we get $e^{-(1-\alpha)} = 1/2M = f(x)$ .Thus it follows

$$f(x) = 1/2M, \quad -M \le x \le M \qquad \text{..... (5.11)}$$

You will immediately identify Eq (5.11) as the uniform density function. Hence the entropy, under peak signal limitation condition, will become a maximum if and only if the signal has a rectangular or uniform distribution. The maximum value of entropy can be found by substituting Eq (5.11) in Eq (5.3) as:

$$H(X)_{max} = \log 2M \text{ bits/sample}. \qquad \text{..... (5.12)}$$

If $S_m = M^2$, then $2M = \sqrt{4 S_m}$ and $H(X)_{max} = \frac{1}{2} \log 4S_m$ **bits/sample.** ..... (5.13)

Suppose the signal is band limited to '***B***' Hz and is sampled at the "Nyquist rate" i.e. ***r=2B samples/sec***., then we have

$$R(X)_{max} = B \log 4S_m \text{ bits/sec.} \qquad \text{..... (5.14)}$$

For the uniform distribution over ***(-M, M)***, we have

Mean = ***0***, and Variance, $\sigma^2 = (2M)^2/12 = M^2/3$

Since the mean value is zero, the variance can be treated as the average power, $\bar{S}$ and since $M^2 = S_m$, it follows $S_m = 3\,\bar{S}$ and Eq (4.14) becomes:

$$R(X)_{max} = B \log 12\,\bar{S} \text{ bits/sec} \qquad \text{..... (5.15)}$$

Case II Average Signal Limitation:

Suppose X has unidirectional distribution with a specified average value (For example: ***PAM***, ***PLM*** or in ***AM*** with average carrier amplitude limitation), the constraint equations are:

$$\int_0^\infty f(x)dx = 1 \qquad \text{(5.16a)}$$

And $\int_0^\infty xf(x)dx = \alpha\ (say)$ .... (5.16b)

The equation to be solved now is $\frac{\partial \varphi}{\partial f} + \alpha_1 \frac{\partial \varphi}{\partial f} + \alpha_2 \frac{\partial \varphi}{\partial f} = 0$

Where $\varphi_1(x, f) = f$ and $\varphi_2(x, f) = x.f$

This leads to $f = e^{-(1-\alpha_1-\alpha_2 x)} = e^{-(1-\alpha_1)}.e^{-ax}$ (5.17)

Where $a = -\alpha_2 > 0$. This step is needed to make the integral converge as you will see later.

Using Eq (5.17) in Eq (5.16a) gives $e^{-(1-\alpha 1)} = a$. Substituting in Eq (5.17) back results in

$$f(x) = ae^{-ax} \qquad \text{(5.18)}$$

Substituting Eq (5.18) in Eq (4.16b) gives $a = 1/\alpha$

Thus the ***p.d.f*** that maximizes ***H(x)*** is

$$f(x) = \frac{1}{\alpha} e^{-\frac{1}{\alpha} x}, \quad x > 0 \qquad \text{(5.19)}$$

Which you will recognize as the exponential density function with parameter $1/\alpha$. The maximum value of the entropy can be shown as:

$H_{max}(X) = \log e + \log \alpha = \log \alpha e$ bits/sample. ..... (5.2 0)

The rate of information transmission over a band width of '***B***', assuming the signal is sampled at the Nyquist rate (i.e. $=2B$ ***samples/sec***.) is:

$$R_{max} = 2B \log \alpha e \text{ bits/sec}. \qquad \text{(5.21)}$$

Case III Average Power Limitation (Symmetrical Distribution):

Examples are Random noise with specified variance, audio frequency telephony and similar situations**.**

The constraint equations assuming zero mean value are now:

$$\int_{-\infty}^{\infty} f(x)dx = 1 \qquad \text{(5.22a)}$$

$$\int_{-\infty}^{\infty} x^2 . f(x)dx = \sigma^2 \quad \text{..... (5.22b)}$$

Hence $\varphi_1(x, f) = f$ and $\varphi_2(x, f) = x^2 . f$

Setting $\frac{\partial \varphi}{\partial f} + \alpha_1 \frac{\partial \varphi_1}{\partial f} + \alpha_2 \frac{\partial \varphi_2}{\partial f} = 0$, we get as a first solution:

$$f = e^{-(1-\alpha_1 - \alpha_2 x_2)} = e^{-(1-\alpha_1)} . e^{-ax_2} \quad \text{..... (5.23)}$$

Where we have set $a = -\alpha_2 > 0$. Substituting Eq (5.23) in Eq (5.22a), the second solution is

$$f = \sqrt{\frac{a}{\pi}} e^{-ax^2} \quad \text{..... .. (5.24)}$$

Use has been made of the formula $\int_{-\infty}^{\infty} e^{-\alpha x^2} dx = \sqrt{\frac{\pi}{\alpha}}$ (From properties of Gamma functions.)Final solution is obtained by substituting Eq (5.24) in Eq (5.22b) and converting the integral into the form of a Gamma integral. Thus we have:

$$\int_{-\infty}^{\infty} x^2 \sqrt{\frac{a}{\pi}} e^{-ax^2} dx = \sigma^2 = 2\sqrt{\frac{a}{\pi}} \int_0^{\infty} x^2 e^{-ax^2} dx$$, as the integrand, $x^2 e^{-ax^2}$, is an even function.

Letting $y = ax^2$ and manipulating, this integral reduces to:

$$\frac{1}{a\sqrt{\pi}} \int_0^{\infty} y^{1/2} e^{-y} dy = \sigma^2 = \frac{1}{a\sqrt{\pi}} \int_0^{\infty} y^{\frac{3}{2}-1} e^{-y} dy = \frac{1}{a\sqrt{\pi}} \Gamma\left(\frac{3}{2}\right) = \frac{1}{2}\Gamma\left(\frac{1}{2}\right) = \frac{1}{2}\sqrt{\pi}$$

Thus $a = 1/2\sigma^2$ and the required density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2} \quad \text{..... .... (5.25)}$$

Eq (5.25) is the Gaussian or Normal density function corresponding to N (0, σ^2) and gives maximum entropy. We have:

$$H(X)_{max} = \int_{-\infty}^{\infty} f(x) \log\{\sigma \sqrt{2\pi}\, e^{x^2/2\sigma^2}\} dx$$

$$= \log \sigma\sqrt{2\pi} \int_{-\infty}^{\infty} f(x)dx + \frac{\log e}{2\sigma^2} \int_{-\infty}^{\infty} x^2 f(x)dx$$

$$= \frac{1}{2} \log 2\pi\sigma^2 + \frac{1}{2} \log e = \frac{1}{2} \log 2\pi e\sigma^2 \text{ bits / sample}$$

In simplifying the integral (not evaluating) use is made of Eq (5.22a) and Eq (5.22b). Thus:

$$H(X)_{max} = (1/2)\log(2\pi e\sigma^2) \text{ bits/sample}. \quad \ldots\ldots\ldots\ldots\ldots (5.26)$$

The rate of information transmission over a band width of '***B***' assuming the signal is sampled at the Nyquist rate (i.e. =***2B***samples/sec.) is:

$$R(X)_{max} = B\log(2\pi e\sigma^2) \text{ bits/sec.} \quad \ldots\ldots\ldots\ldots\ldots (5.27)$$

Since 'σ^2' represents the signal power, ***S***, Eq (5.27) can also be expressed as below:

$$R(X)_{max} = B\log(2\pi e S) \text{ bits/sec} \quad \ldots\ldots\ldots\ldots\ldots (5.28)$$

If '***X***' represents white Gaussian noise (***WGN***), with an average power $\sigma^2 = N$, Eq (4.28) becomes:

$$R(X)_{max} = B\log(2\pi e N) \text{ bits/sec} \quad \ldots\ldots\ldots\ldots\ldots (5.29)$$

Case IV: Average Power Limitation (Unidirectional Distribution):

If the signal ***X*** has a unidirectional distribution with average power limitation (***AM*** with average carrier power constraint), the constraint equations become:

$$\int_{-\infty}^{\infty} f(x)dx = 1 \text{ , and} \qquad \int_{0}^{\infty} x^2 . f(x)dx = P_0$$

Then following the steps, as in other cases, you can show that:

$$f(x) = \sqrt{\frac{2}{\pi P_0}}\, exp\left(-x^2/2P_0\right), \quad 0 \le x \le \infty \quad \ldots\ldots\ldots\ldots\ldots (5.30)$$

$$H(X)_{max} = \frac{1}{2}\log\frac{\pi e P_0}{2} \text{ bits/sample} \quad \ldots\ldots\ldots\ldots\ldots (5.31)$$

$$R(X)_{max} = B\log\frac{\pi e P_0}{2} \text{ bits/sec.} \quad \ldots\ldots\ldots\ldots\ldots (5.32)$$

Compared to Eq (5.28), it is clear that the entropy in Eq (5.32) is smaller by ***2 bits/sec***.

NOTE*:** In case III we have shown that the entropy of Gaussian signals is $\log\sqrt{2\pi e\sigma^2} = \log\sqrt{2\pi eS}$, where 'S***' is the signal power and is maximum among all *p.d.f* `s of continuous signals with average power limitation. Often times, for the calculation of the transmission rate and channel capacity, it is convenient to find equivalent entropy of signals and interference which are neither random nor Gaussian. We define "***Entropy Power***" S_e of an ensemble of samples limited to the same band width, ***B***, and period ***T*** as the original ensemble, and having the same entropy ***H*** bits/sample. Then ***H***/sample= $\log\sqrt{2\pi eS_e}$ .Therefore $S_e = e^{2H}/2\pi e$. Since ***H*** for a random signal is maximum for a given ***S*** when it is Gaussian, then S_e for any arbitrary signal is less than or equal to the average power of the signal.

Mutual Information of a Continuous Noisy Channel:

While remembering that the entropy definition for continuous signals is a relative one and using the logarithmic property, it follows that all entropy relations that we have studied for discrete signals do hold for continuous signals as well. Thus we have

$$H(X,Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\log\frac{1}{f(x,y)}dxdy \quad \ldots\ldots (5.33)$$

$$H(X|Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\log\frac{1}{f(x|y)}dxdy \quad \ldots\ldots\ldots (5.34)$$

The mutual information is $I(X, Y) = H(X) - H(X|Y)$ …… (5.35)

$$= H(Y) - H(Y|X) \quad \ldots\ldots (5.36)$$

$$= H(X) + H(Y) - H(X, Y) \quad \ldots\ldots (5.37)$$

The channel capacity is $C = Max\ I(X, Y)$ …… (5.38)

And so on. You can easily verify the various relations among entropies.

Amount of Mutual Information:

Suppose that the channel noise is additive, and statistically independent of the transmitted signal. Then, $f(y|x)$ depends on $(y-x)$, and not on x or y. Since $Y=X+n$, where, n is the channel noise, and $f(y|x) = f_Y(y|X=x)$.

It follows that when X has a given value, the distribution of Y is identical to that of n, except for a translation of X. If $f_n(.)$ represents the *p.d.f.* of noise sample, n, then obviously $f_Y(y|x) = f_n(y-x)$

$$\therefore \int_{-\infty}^{\infty} f_Y(y|x)\log\frac{1}{f_Y(y|x)}dy = \int_{-\infty}^{\infty} f_n(y-x)\log\frac{1}{f_n(y-x)}dy$$

Letting $z = y - x$, we have then

$$H(Y|x) = \int_{-\infty}^{\infty} f_n(z)\log\frac{1}{f_n(z)}dz = H(n)$$

Accordingly,

$$H(Y|X) = \int_{-\infty}^{\infty} f_X(x)H(Y|x)dx = \int_{-\infty}^{\infty} f_X(x)H(n)dx = H(n)\int_{-\infty}^{\infty} f_X(x)dx = H(n)$$

Or $H(Y|X) = H(n)$ … (5.39)

There fore, ***the amount of mutual information*** $= H(Y) - H(Y|X)$

$$That\ is,\ I(X, Y) = H(Y) - H(n) \quad \ldots\ldots (5.40)$$

The equivocation is H $(X|Y) = H(X) - I(X, Y) = H(X) - H(Y) + H(n)$ …… (5.41)

Since the Channel capacity is defined as

$$C = R_{max} = 2B\, I(X, Y)_{max} \text{ bits/sec.} = [R(Y) - R(Y|X)]_{max}$$

It follows: $C = [R(Y) - R(n)]_{max}$ ..... …….. (5.42)

Capacity of band limited channels with AWGN and Average Power

Limitation of signals: The Shannon – Hartl ey law:

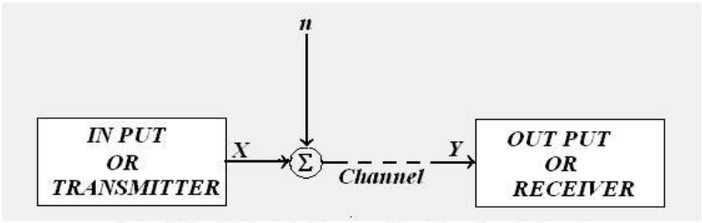


Fig 5.1 A Simple Transmitter-Receiver Structure

Consider the situation illustrated in Fig. 5.1. The received signal will be composed of the transmitted signal ***X*** plus noise ' ***n***'. The joint entropy at the transmitter end, assuming signal and noise are independent, is:

$$H(X, n) = H(X) + H(n|X).$$

$$= H(X) + H(n) \quad \text{……(5.43)}$$

The joint entropy at the receiver end, however, is

$$H(X, Y) = H(Y) + H(X|Y) \quad \text{…… (5.44)}$$

Since the received signal is $Y = X + n$, and the joint entropy over the channel is invariant, it follows:

$$H(X, n) = H(X, Y) \quad \text{…… (5.45)}$$

The, from Eq. (5.43) and Eq (5.44), one obtains

$$I(X, Y) = H(X) - H(X \mid Y)$$

$$= H(Y) - H(n) \quad \text{.....(5.46)}$$

Alternatively, we could have directly started with the above relation in view of Eq. (5.40). Hence, it follows, the channel capacity, in bits / second is:

$$C = \{R(Y) - R(n)\}_{max} \quad \text{..... (5.47)}$$

If the additive noise is white and Gaussian, and has a power ' ***N***' in a bandwidth of ' ***B***' Hz, then from Eq. (5.29), we have:

$$R(n)_{max} = B \log 2\pi eN \quad \text{..... (5.48)}$$

Further, if the input signal is also limited to an average power S over the same bandwidth, and ***X*** and ***n*** are independent then it follows:

$$\sigma_Y^2 = (S + N)$$

We have seen that for a given mean square vale, the entropy will become a maximum if the signal is Gaussian, and there fore the maximum entropy of the output is:

$$H(Y)_{max} = (1/2) \log 2\pi e (S + N) \text{ bits/sample}$$

Or, $$R(Y)_{max} = B \log 2\pi e (S + N) \text{ bits/sec} \quad \text{.....(5.49)}$$

If ' ***n***' is an ***AWGN***, then ***Y*** will be Gaussian if and only if, ***X*** is also Gaussian. This implies

$$f_X(x) = \frac{1}{2\pi} \exp \frac{x^2}{2S}$$

Using Eq. (5.48) and (5.49) in Eq. (5.47), one obtains

$$C = B \log \left(1 + \frac{S}{N}\right) \text{ bits/sec} \quad \text{.....(5.50)}$$

Or, $$C = B \log (1 + \Lambda) \text{ bits/sec} \quad \text{.....(5.51)}$$

Where, $\Lambda = S/N$, is the signal to noise power ratio.

This result (Eq. 5.50) is known as " ***Shannon-Hartley law***". The primary significance of the formula is that it is possible to transmit over a channel of bandwidth ***B*** Hz perturbed by ***AWGN*** at a rate of ***C*** bits/sec with an arbitrarily small probability of error if the signal is encoded in such a manner that the samples are all Gaussian signals. This can, however, be achieved by Orthogonal codes.

Bandwidth – SNR Tradeoff:

One important implication of the 'Shannon - Hartley Law' is the exchange of bandwidth with signal to noise power ratio. Suppose, ***S/N = 7***, and ***B = 4KHz*** then from Eq. (4.50), ***C = 12000 bits/sec***. Let ***S/N*** be increased to ***15***, while the bandwidth is reduced to ***3 KHz***. We see that the channel capacity remains the same. The Power Spectral Density, $S_n(f)$, of White Gaussian noise is more or less a constant over the entire frequency range,$(-\infty, \infty)$, with a two sided ***p.s.d*** of $N_0/2$ as shown in Fig 5.3.

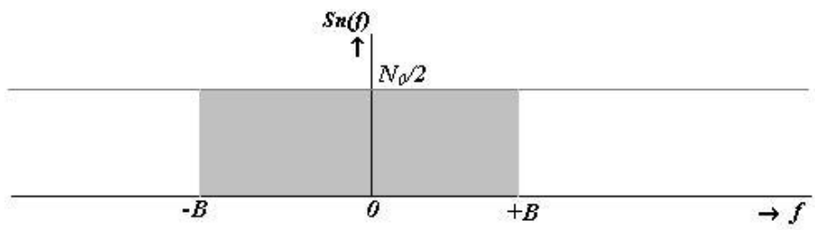


Fig 5.3 Power Spectral Density Of White Gaussian Noise

From the figure we find the noise power over (-***B***, ***B***) as $N = N_0 / 2 \,.2B$ or

$$N = N_0 B \qquad \text{.....(5.52)}$$

That is, the noise power is directly proportional to the band width ***B***. Thus the noise power will be reduced by reducing the band width and vice-versa. This indicates, then, an increase in the signal power and vice-versa. For illustration consider the following:

With $\frac{S_1}{N_1} = 7$ and $B_1 = 4\ KHz$, we have from Eq (5.50) $C = 12000\ bits/sec$.

Suppose we decrease the band width by ***25%*** and choose the new values of ***SNR*** and band width as $\frac{S_2}{N_2} = 15$ and $B_2 = 3\ KHz$, we get the same channel capacity as before.

Then $\frac{S_1}{N_1} \div \frac{S_2}{N_2} = 7/15$

Using Eq (5.52), this means

$$\frac{S_1}{S_2} = \frac{7}{15} \cdot \frac{B_1}{B_2} = \frac{7}{15} \cdot \frac{4}{3} = \frac{28}{45} = 0.62222 \text{ or } S_2 = \frac{45}{28} S_1 = 1.607\, S_1$$

Thus a ***25 %*** reduction in bandwidth requires a ***60.7 %*** increase in signal power for maintaining the same channel capacity. We shall inquire into this concept in a different way. Eq. (5.50) can be written in the form:

$$\frac{B}{C} \log_2 \left(1 + \frac{S}{N}\right) = 1 \qquad \text{.....(5.53)}$$

Table below shows the variation of (***B/C***) with (***S/N***)

S/N	0.5	1	2	5	10	15	20	30
B/C	1.71	1.0	63	0.37	0.289	0.25	0.23	0.20

A plot of (***B/C***) versus (***S/N*** is shown in Fig 5.4. Clearly, the same channel capacity may be obtained increasing ***S/N*** is poor. Use of larger band width for smaller ***S/N*** is generally known as " *coding upwards*" and use of smaller ***B*** with larger ***S/N*** is called " *coding downwards*". One can consider as examples of coding upwards, the ***FM***, ***PM*** and ***PCM*** systems where larger band widths are used with improvements in ***S/N*** ratio. Quantization of the signal samples and then, combining the different sample values into a single pulse, as in the case of multi-level, discrete ***PAM*** can be considered as an example of coding downwards where the band width reduction depends on the signal power available.

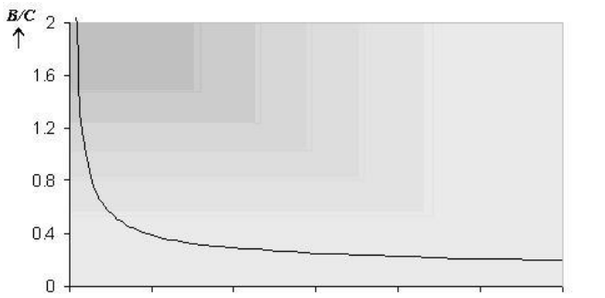


Fig 5.4 Band Width - SNR Trade Off Graph

For wide band systems, where (***S/N***) **>> 1**, use of Eq (5.50) leads to:

$$C = B_1 \log_{1+} \frac{S_1}{N_1} = B_2 \log_{} 1+\frac{S_2}{N_2}$$

$$or \quad 1+\frac{S_2}{N_2} = 1+\frac{S_1}{N_1}^{B_1/B_2}$$

$$Or \quad \frac{S_2}{N_2} = \frac{S_1}{N_1}^{B_1/B_2}, \text{ when } \frac{S_1}{N_1} \text{ and } \frac{S_2}{N_2} >> 1 \quad \ldots\ldots\ldots\ldots \quad (5.54)$$

Notice that Eq. (5.54) predicts an exponential improvement in (***S/N***) ratio with band width for an ideal system. For the conventional demodulation methods used in FM and PM, however, the (***S/N***) ratio varies as the square of the transmission bandwidth, which, obviously, is inferior to the ideal performance indicated by Eq. (5.54). However, better performance that approaches the Shannon-bound in the limit as bandwidth is made infinite, can be obtained by using optimum demodulators (Phase-locked loop).

Capacity of a Channel of Infinite Bandwidth:

The Shannon-Hartley formula predicts that a noiseless Gaussian channel with (***S/N*** = ∞) has an infinite capacity. However, the channel capacity does not become infinite when the bandwidth is made infinite, in view of Eq. (5.52). From Eq. (5.52), Eq. (5.50) is modified to read:

$$C = \lim_{B \to \infty} 1 + \frac{S}{N_0 B} \text{ bits/sec.}$$

$$= \frac{S}{N_0} \cdot \frac{N_0 B}{S} \log_{1} + \frac{S}{N_0 B}$$

$$= \frac{S}{N_0} \cdot \frac{1}{x} \log(1+x) \ldots. \; Where \; x = \frac{S}{N_0 B}$$

Or $$C = \frac{S}{\log(1+x)_x N_0}^{\frac{1}{}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (5.55)$$

Accordingly, when $B \to \infty$, $x \to 0$ and we have

$$C_{max} = \lim_{B \to \infty} C = \frac{S}{N_0} \log \lim_{x \to 0} (1+x)^{\frac{1}{x}} \quad \text{..... (5.56)}$$

Since $\lim_{x \to 0} (1+\alpha x)^{1/x} = e^{\alpha}$ and $\log e = 1.442685$, we have

$$C_{max} = 1.442685 \frac{S}{N_0} \text{ bits/sec} \quad \text{..... (5.57)}$$

Eq (5.57) places an upper limit on channel capacity with increasing band width.

Bandwidth-Efficiency: Shannon Limit:

In practical channels, the noise power spectral density N_0 is generally constant. If E_b is the transmitted energy per bit, then we may express the average transmitted power as:

$$S = E_b C \quad \text{..... .. (5.58)}$$

Using Eq. (5.52) and (5.58), Eq. (5.50) may now be re-formulated as:

$$\frac{C}{B} = 1 + \frac{E_b}{N_0} \cdot \frac{C}{B} \quad \text{..... . (5.59)}$$

From which one can show:

$$\frac{E_b}{N_0} = \frac{2^{C/B} - 1}{\frac{C}{B}} \quad \text{..... (5.60)}$$

(C/B) is called the "bandwidth efficiency" of the syste m. If $C/B = 1$, then it follows that $E_b = N_0$.This implies that the signal power equals the noise power. Suppose, $B = B_0$ for which, $S = N$, then Eq. (5.59) can be modified as:

$$\frac{C}{B_0} = \frac{B}{B_0} \log \left(1 + \frac{B}{B}\right) \quad \text{... (5.61)}$$

And

$$\lim_{B \to \infty} \frac{C}{B_0} = \frac{1}{B_0} \lim_{B \to \infty} \log \left(1 + \frac{B_0}{B}\right)^{\frac{B}{B_0}} = \frac{1}{B_0} \log e$$

$$\therefore C_{max} = B_0 \log e \quad \text{..... (5.62)}$$

That is, " *the maximum signaling rate for a given* S *is 1.443 bits/sec/Hz in the bandwidth over which the signal power can be spread without its falling below the noise level*".

It follows from Eq. (5.58):

$$S = E_b C = N_0 B_0 \quad \text{or} \quad E_b/N_0 = B_0/C$$

And as $B \to \infty$, we have And as $B \to \infty$, we have

$$\lim_{B \to \infty} \frac{E_b}{N_0} = \lim_{B \to \infty} \frac{B}{C} = \frac{B}{C_{max}} = \frac{B}{B_0 \log e} = \ln 2$$

In other words

$$\left(\frac{E_b}{N_0}\right)_{min} = \ln 2 = 0.69314718 \quad \text{..... (5.63)}$$

Expressed in decibels, this means:

$$\left(\frac{E_b}{N_0}\right)_{min} = 10 \log 0.69314718 \text{ dB} = -1.59174539 \text{ dB} \quad \text{..... (5.64)}$$

The value $\left(\frac{E_b}{N_0}\right)_{min} = -1.592$ dB is known as the "***Shannon`s limit***" for transmission at capacity $C_{max} = 1.443S/N_0$ and the communication fails otherwise (i.e. $E_b/N_0 < 0.693$.).

We define an "ideal system" as one that transmits d ata at a bit rate R equal to the channel capacity. Fig 5.5 shows a plot of the bandwidth efficiency, $R / B = C/B = \eta_b$, as a

function of the "energy-per-bit to noise power spec tral density ratio" E_b/N_0. Such a diagram is called "bandwidth efficiency diagram". Clearly shown on th e diagram are:

- The capacity boundary defined by the curve $R = C$.
- Region for which $R > C$ shown as dark gray area for which error-free transmission is not possible.
- Region for which $R < C$ (light gray portion) in which combinations of system-parameters have the potential for supporting error-free transmission.

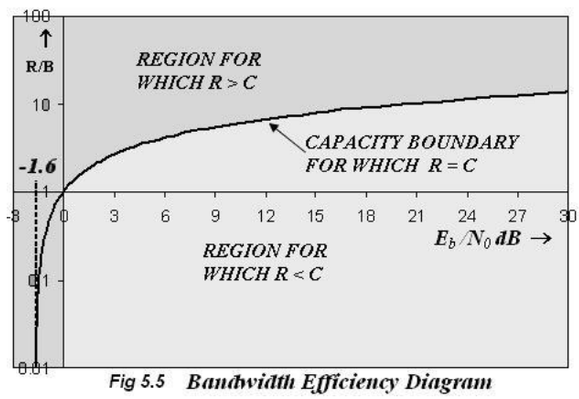


Fig 5.5 Bandwidth Efficiency Diagram

Example 5.3

A vice-grade channel of a telephone network has a bandwidth of 3.4 kHz.

a) *Calculate the channel capacity of the telephone channel for a signal to noise ratio of 30dB.*

b) *Calculate the minimum signal to noise ratio required to support information transmission through the telephone channel at the rate of 4800 bits/sec.*

c) Solution:

a) B = 3.4kHz, S/N = 30dB

$[S/N]_{dB} = 10 \log_{10}[S/N]_{Abs}$ (Remember ***S/N*** is a Power ratio)

$\therefore S/N = 10^{\{[S/N]\ dB/10\}} = 10^3 = 1000$

$\therefore C = B \log_2\{1 + S/N\} = 33888.56928$ ***bits/sec***

b) C = 4800, B =3.4 kHz and S/N = $(2^{C/B} - 1) = 1.660624116$ or 2.2 dB

Example 5.4

A communication system employs a continuous source. The channel noise is white and Gaussian. The bandwidth of the source output is 10 MHz and signal to noise power ratio at the receiver is 100.

a) *Determine the channel capacity*

b) *If the signal to noise ratio drops to 10, how much bandwidth is needed to achieve the same channel capacity as in (a).*

c) *If the bandwidth is decreased to 1MHz, what S/N ratio is required to maintain the same channel capacity as in (a).*

Solution:

a) $C = 10^7 \log(101) = 6.66 \times 10^7$ ***bits/sec***

b) $B = C / \log_2(1 + S/N) = C/\log_2 11 = 19.25$ ***MHz***

c) $S/N = (2^{C/B} - 1) = 1.105 \times 10^{20} = 200.43$***dB***

Example 5.5:

Alphanumeric data are entered into a computer from a remote terminal though a voice grade telephone channel. The channel has a bandwidth of 3.4 kHz and output signal to noise power ratio of 20 dB. The terminal has a total of 128 symbols which may be assumed to occur with equal probability and that the successive transmissions are statistically independent.

a) *Calculate the channel capacity:*

b) *Calculate the maximum symbol rate for which error-free transmission over the channel is possible.*

Solution:

a) $S/N = 10^{20/10} = 10^2 = 100$

$C = 3.4 \times 10^3 \log_2 (1 + 100) = 22637.92$ bits/sec

b) $H_{max} = \log_2 128 = 7$ bits per symbol

$C = r_{s\,max} \cdot H_{max}$

Therefore, $r_{s\,max} = C/H_{max} = 3233.99$ bits/sec $= 3.234$ kb/sec

Example 5.6

A black and white television picture may be viewed as consisting of approximately 3 x10^5 elements, each one of which may occupy one of 10 distinct brightness levels with equal probability. Assume the rate of transmission to be 30 picture frames per second, and the signal to noise power ratio is 30 dB.

Using the channel capacity theorem, calculate the minimum bandwidth required to support the transmission of the resultant video signal.

Solution:

$(S/N)_{dB} = 30$ dB or $(S/N)_{Abs} = 1000$

No. of different pictures possible = $10^{3 \times 105}$

Therefore, entropy per picture $= 3 \times 10^5 \log_2 10 = 9.965784285 \times 10^5$ bits

$\therefore$ Entropy rate = $r_s \cdot H = 30 \times H = 298.97 \times 10^5$ bits/sec

$C = r_s \cdot H = B \log_2 [1+S/N]$

$\therefore B_{min} = r_s \cdot H \div B \log_2 [1+S/N] = 3.0$ MHz

Note: As a matter of interest, commercial television transmissions actually employ a bandwidth of ***4 MHz.***

Review questions:

1. Show that for a AWGN channel $C_\infty = \frac{1.448}{\eta}$ where $\eta/2$ = noise power spectral density in watts/Hz.

2. Consider an AWGN channel with 4 KHz bandwidth with noise power spectral density $\frac{\eta}{2} = 10^{-12}$ watts/Hz. The signal power required at the receiver is 0.1mW. Calculate the capacity of the channel.

3. If $I(x_i, y_i) = I(x_i)-I(x_i/y_j)$. Prove that $I(x_i,y_j)=I(y_j)-I(y_j/x_i)$

4. Consider a continuous random variable having a distribution as given below

5. $F_X(X) = A$ $\begin{cases} \frac{1}{-} & 0 \le X \le A \\ 0 & \text{OTHERWISE} \end{cases}$

6. Find the differential entropy H(x)

7. Design a single error correcting code with a message block size of 11 and show that by an example that it can correct single error.

8. If C_i and C_j an two code vectors in a (n,k) linear block code, show that their sum is also a code vector.

9. Show $CH^T=0$ for a linear block code.

10. Prove that the minimum distance of a linear block code is the smallest weight of the non-zero code vector in the code.

.

PART B

UNIT 5

CONCEPTS OF ERROR CONTROL CODING -- BLOCK CODES

Syllabus:

Introduction, Types of errors, examples, Types of codes Linear Block Codes: Matrix description, Error detection and correction, Standard arrays and table look up for decoding.

7 Hours

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996. Digital communication, Simon Hay kin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007

Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

UNIT 5

CONCEPTS OF ERROR CONTROL CODING -- BLOCK CODES

The earlier chapters have given you enough background of Information theory and Source encoding. In this chapter you will be introduced to another important signal - processing operation, namely, " ***Channel Encoding***", which is used to provide 'reliable' transmission of information over the channel. In particular, we present, in this and subsequent chapters, a survey of 'Error control coding' techniques that rely on the systematic addition of 'Redundant' symbols to the transmitted information so as to facilitate two basic objectives at the receiver: 'Error- detection' and 'Error correction'. We begin with some preliminary discussions highlighting the role of error control coding.

5.1 Rationale for Coding:

The main task required in digital communication is to construct 'cost effective systems' for transmitting information from a sender (one end of the system) at a rate and a level of reliability that are acceptable to a user (the other end of the system). The two key parameters available are transmitted signal power and channel band width. These two parameters along with power spectral density of noise determine the signal energy per bit to noise power density ratio, E_b/N_0 and this ratio, as seen in chapter 4, uniquely determines the bit error for a particular scheme and we would like to transmit information at a rate $R_{Max} = 1.443\ S/N$. Practical considerations restrict the limit on E_b/N_0 that we can assign. Accordingly, we often arrive at modulation schemes that cannot provide acceptable data quality (i.e. low enough error performance). For a fixed E_b/N_0, the only practical alternative available for changing data quality from problematic to acceptable is to use "coding".

Another practical motivation for the use of coding is to reduce the required E_b/N_0 for a fixed error rate. This reduction, in turn, may be exploited to reduce the required signal power or reduce the hardware costs (example: by requiring a smaller antenna size).

The coding methods discussed in chapter 5 deals with minimizing the average word length of the codes with an objective of achieving the lower bound viz. ***H(S) / log r***, accordingly, coding is termed "entropy coding". However, such source codes cannot be adopted for direct transmission over the channel. We shall consider the coding for a source having four symbols with probabilities $p(s_1) = 1/2$, $p(s_2) = 1/4$, $p(s_3) = p(s_4) = 1/8$. The resultant binary code using Huffman's procedure is:

s_1..... **0** **s** $_3$..... **1 1 0**

s_2..... **10** **s** $_4$..... **1 1 1**

Clearly, the code efficiency is ***100%*** and ***L = 1.75 bints/sym = H(S)***. The sequence $s_3s_4s_1$ will then correspond to ***1101110***. Suppose a one-bit error occurs so that the received sequence is ***0101110***. This will be decoded as " $s_1s_2s_4s_1$", which is altogether different than the transmitt ed sequence. Thus although the coding provides ***100%*** efficiency in the light of Shannon's theorem, it suffers a major disadvantage. Another disadvantage of a ' ***variable length***' code lies in the fact that output data rates measured over short time periods will fluctuate widely. To avoid this problem, buffers of large length will be needed both at the encoder and at the decoder to store the variable rate bit stream if a fixed output rate is to be maintained.

Some of the above difficulties can be resolved by using codes with " ***fixed length***". For example, if the codes for the example cited are modified as ***000***, ***100***, ***110***, and ***111***. Observe that even if there is a one-bit error, it affects only one " ***block***" and that the output data rate will not fluctuate.

The encoder/decoder structure using ' ***fixed length***' code words will be very simple compared to the complexity of those for the variable length codes.

Here after, we shall mean by " ***Block codes***", the fixed length codes only. Since as discussed above, single bit errors lead to ' ***single block errors***', we can devise means to detect and correct these errors at the receiver. Notice that the price to be paid for the efficient handling and easy manipulations of the codes is reduced efficiency and hence increased redundancy.

In general, whatever be the scheme adopted for transmission of digital/analog information, the probability of error is a function of signal-to-noise power ratio at the input of a receiver and the data rate. However, the constraints like maximum signal power and bandwidth of the channel (mainly the Governmental regulations on public channels) etc, make it impossible to arrive at a signaling scheme which will yield an acceptable probability of error for a given application. The answer to this problem is then the use of ' ***error control coding***', also known as ' ***channel coding***'. In brief, ***"error control coding is the calculated addition of redundancy"*** . The block diagram of a typical data transmission system is shown in Fig. 6.1

The information source can be either a person or a machine (a digital computer). The source output, which is to be communicated to the destination, can be either a continuous wave form or a sequence of discrete symbols. The ' ***source encoder***' transforms the source output into a sequence of binary digits, the information sequence ***u***. If the source output happens to be continuous, this involves ***A-D*** conversion as well. The source encoder is ideally designed such that (i) the number of bints per unit time (bit rate, r_b) required to represent the source output is minimized (ii) the source output can be uniquely reconstructed from the information sequence ***u***.

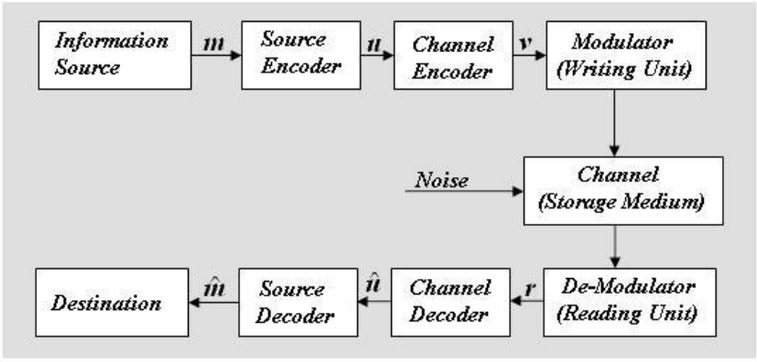


Fig 6.1 Block diagram of a Typical data transmission system.

The ' ***Channel encoder***' transforms ***u*** to the encoded sequence ***v***, in general, a binary sequence, although non-binary codes can also be used for some applications. As discrete symbols are not suited for transmission over a physical channel, the code sequences are transformed to waveforms of specified durations. These waveforms, as they enter the channel get corrupted by noise. Typical channels include telephone lines, High frequency radio links, Telemetry links, Microwave links, and Satellite links and so on. Core and semiconductor memories, Tapes, Drums, disks, optical memory and so on are typical storage mediums. The switching impulse noise, thermal noise, cross talk and lightning are some examples of noise disturbance over a physical channel. A surface defect on a magnetic tape is a source of disturbance. The demodulator processes each received waveform and produces an output, which may be either continuous or discrete – the sequence ***r***. The channel

decoder transforms ***r*** into a binary sequence, $\hat{u}$ which gives the estimate of ***u***, and ideally should be

the replica of ***u***. The source decoder then transforms ***u^*** into an estimate of source output and delivers this to the destination.

Error control for data integrity may be exercised by means of ' ***forward error correction***' (***FEC***) where in the decoder performs ***error correction*** operation on the received information according to the schemes devised for the purpose. There is however another major approach known as ' ***Automatic Repeat Request***' (***ARQ***), in which a re-transmission of the ambiguous information is effected, is also used for solving error control problems. In ***ARQ***, error correction is not done at all. The redundancy introduced is used only for ' ***error detection***' and upon detection, the receiver requests a repeat transmission which necessitates the use of a return path (feed back channel).

In summary, channel coding refers to a class of signal transformations designed to improve performance of communication systems by enabling the transmitted signals to better withstand the effect of various channel impairments such as noise, fading and jamming. Main objective of error control coding is to reduce the probability of error or reduce the E_b/N_0 at the cost of expending more bandwidth than would otherwise be necessary. Channel coding is a very popular way of providing performance improvement. Use of ***VLSI*** technology has made it possible to provide as much as ***8 – dB*** performance improvement through coding, at much lesser cost than through other methods such as high power transmitters or larger Antennas.

We will briefly discuss in this chapter the channel encoder and decoder strategies, our major interest being in the design and implementation of the channel ' ***encoder/decoder***' pair to achieve fast transmission of information over a noisy channel, reliable communication of information and reduction of the implementation cost of the equipment.

5.2 Types of errors:

The errors that arise in a communication system can be viewed as ' ***independent errors***' and ' ***burst errors***'. The first type of error is usually encountered b y the ' ***Gaussian noise***', which is the chief concern in the design and evaluation of modulators and demodulators for data transmission. The possible sources are the thermal noise and shot noise of the transmitting and receiving equipment, thermal noise in the channel and the radiations picked up by the receiving antenna. Further, in majority situations, the power spectral density of the Gaussian noise at the receiver input is white. The transmission errors introduced by this noise are such that the error during a particular signaling interval does not affect the performance of the system during the subsequent intervals. The discrete channel, in this case, can be modeled by a Binary symmetric channel. These transmission errors due to Gaussian noise are referred to as ' ***independent errors***' (***or random errors***).

The second type of error is encountered due to the ' ***impulse noise***', which is characterized by long quiet intervals followed by high amplitude ***noise bursts*** (As in switching and lightning). A noise burst usually affects more than one symbol and there will be dependence of errors in successive transmitted symbols. Thus errors occur in bursts

5. 3 Types of codes:

There are mainly two types of error control coding schemes – ***Block codes*** and ***convolutional codes***, which can take care of either type of errors mentioned above.

In a block code, the information sequence is divided into message blocks of ***k*** bits each, represented by a binary ***k***-tuple, $\boldsymbol{u} = (u_1, u_2 \ldots u_k)$ and each block is called a message. *The symbol* ***u****, here, is used to denote a* ***k*** *– bit message rather than the entire information sequence* . The encoder

then transforms $\boldsymbol{u}$ into an $\boldsymbol{n}$-tuple $\boldsymbol{v} = (v_1, v_2 \ldots v_n)$. *Here* $\boldsymbol{v}$ *represents an encoded block rather than the entire encoded sequence*. The blocks are independent of each other.

The encoder of a convolutional code also accepts $\boldsymbol{k}$-bit blocks of the information sequence $\boldsymbol{u}$ and produces an $\boldsymbol{n}$-symbol block $\boldsymbol{v}$. *Here* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *are used to denote sequences of blocks rather than a single block.* Further each encoded block depends not only on the present $\boldsymbol{k}$-bit message block but also on $\boldsymbol{m}$-pervious blocks. Hence the encoder has a memory of order ' $\boldsymbol{m}$'. Since the encoder has memory, implementation requires sequential logic circuits.

If the code word with $\boldsymbol{n}$-bits is to be transmitted in no more time than is required for the transmission of the $\boldsymbol{k}$-information bits and if τ_b and τ_c are the bit durations in the encoded and coded words, i.e. the input and output code words, then it is necessary that

$$n.\tau_c = k.\tau_b$$

We define the " ***rate of the code***" by (also called ***rate efficiency***)

$$R_c \triangleq \frac{k}{n}$$

Accordingly, with $f_b = \frac{1}{\tau_b}$ and $f_c = \frac{1}{\tau_c}$, we have $\frac{f_b}{f_c} = \frac{\tau_c}{\tau_b} = \frac{k}{n} = R_c$

5.4 Example of Error Control Coding:

Better way to understand the important aspects of error control coding is by way of an example. Suppose that we wish transmit data over a telephone link that has a useable bandwidth of ***4 KHZ*** and a maximum ***SNR*** at the out put of ***12 dB***, at a rate of ***1200 bits/sec*** with a probability of error less than 10^{-3}. Further, we have ***DPSK*** modem that can operate at speeds of ***1200***, ***1400*** and ***3600 bits/sec*** with error probabilities $2\times(10^{-3})$, $4\times(10^{-3})$ and $8\times(10^{-3})$ respectively. We are asked to design an error control coding scheme that would yield an overall probability of error $< 10^{-3}$. We have:

$C = 16300$ ***bits/sec***, $R_c = 1200$, ***2400*** or ***3600 bits/sec.***

[$C = B\log_2(1+\frac{S}{N})$. $\frac{S}{N} = 12dB$ *or* 15.85 , $B=4KHZ$], $p = 2(10^{-3})$, $4(10^{-3})$ and $8(10^{-3})$ respectively. Since $R_c < C$, according to Shannon's theorem, we should be able to transmit data with arbitrarily small probability of error. We shall consider two coding schemes for this problem.

1. ***Error detection***: Single parity check-coding. Consider the (***4, 3***) even parity check code.

Message	*000*	*001*	*010*	*011*	*100*	*101*	*110*	*111*
Parity	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*
Codeword	*0000*	*0011*	*0101*	*0110*	*1001*	*1010*	*1100*	*1111*

Parity bit appears at the right most symbol of the codeword.

This code is capable of ' ***detecting***' all single and triple error patterns. Data comes out of the channel encoder at a rate of ***3600 bits/sec*** and at this rate the modem has an error probability of $8\times(10^{-3})$. The decoder indicates an error only when parity check fails. This happens for single and triple errors only.

p_d = Probability of error detection.

$= p(X=1) + p(X=3)$, where X = Random variable of errors.

Using binomial probability law, we have with $p = 8(10^{-3})$:

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$p_d = \binom{4}{1} p(1-p)^3 + \binom{4}{3} p^3 (1-p), \quad \binom{4}{1} = 4C_1 = 4, \quad \binom{4}{3} = 4C_3 = 4$$

Expanding we get $p_d = 4p - 12p^2 + 16p^3 - 8p^4$

Substituting the value of p we get:

$$p_d = 32\times(10^{-3}) - 768\times(10^{-6}) + 8192\times(10^{-9}) - 32768\times(10^{-12}) = 0.031240326 >> (10^{-3})$$

However, an error results if the decoder does not indicate any error when an error indeed has occurred. This happens when ***two*** or ***4*** errors occur. Hence probability of a detection error = p_{nd} (probability of no detection) is given by:

$$p_{nd} = P(X=2) + P(X=4) = \binom{4}{2} p^2 (1-p)^2 + \binom{4}{4} p^4 (1-p)^0 = 6p^2 - 12p^3 + 7p^4$$

Substituting the value of p we get $p_{nd} = 0.4\times10^{-3} < 10^{-3}$

Thus probability of error is less than 10^{-3} as required.

2. ***Error Correction:*** The triplets ***000*** and ***111*** are transmitted whenever ***0*** and ***1*** are inputted. A majority logic decoding, as shown below, is employed assuming only single errors.

Received Triplet	***000***	***001***	***010***	***100***	***011***	***101***	***110***	***111***
Output message	***0***	***0***	***0***	***0***	***1***	***1***	***1***	***1***

Probability of decoding error, p_{de}= ***P*** (two or more bits in error)

$$\binom{3}{2} p^2 \qquad (1-p) + \binom{3}{3} p^3 (1-p)^0 = 3p^2 - 2p^3$$

$$= 190.464 \times 10^{-6} = 0.19 \times 10^{-3} < p = 10^{-3}$$

Probability of no detection, p_{nd} = P (All 3 bits in error) $= p^3 = 512 \times 10^{-9} << p_{de}$!

In general observe that probability of no detection, p_{nd} < < probability of decoding error, p_{de}.

The preceding examples illustrate the following aspects of error control coding. Note that in both examples with out error control coding the probability of error $= 8 \times (10^{-3})$ of the modem.

1. It is possible to detect and correct errors by adding extra bits-the check bits, to the message sequence. Because of this, not all sequences will constitute bonafide messages.

2. It is not possible to detect and correct all errors.

3. Addition of check bits reduces the effective data rate through the channel.

4. Since probability of no detection is always very much smaller than the decoding error probability, it appears that the error detection schemes, which do not reduce the rate efficiency as the error correcting schemes do, are well suited for our application. Since error detection schemes always go with ***ARQ*** techniques, and when the speed of communication becomes a major concern, Forward error correction (***FEC***) using error correction schemes would be desirable.

5.5 Block codes:

We shall assume that the output of an information source is a sequence of Binary digits. In ' *Block coding*' this information sequence is segmented into ' *message*' blocks of fixed length, say ***k***. Each message block, denoted by ***u*** then consists of ***k*** information digits. The encoder transforms these ***k***-tuples into blocks of code words ***v***, each an ***n***- tuple 'according to certain rules'. Clearly, corresponding to 2^k information blocks possible, we would then have 2^k code words of length $n > k$. This set of 2^k code words is called a " ***Block code***". For a block code to be useful these 2^k code words must be distinct, i.e. there should be a one-to-one correspondence between ***u*** and ***v***. ***u*** and ***v*** are also referred to as the ' ***input vector***' and ' ***code vector***' respectively. Notice that encoding equipment must be capable of storing the 2^k code words of length $n > k$. Accordingly, the complexity of the equipment would become prohibitory if ***n*** and ***k*** become large unless the code words have a special structural property conducive for storage and mechanization. This structural is the ' ***linearity***'.

5.5.1 Linear Block Codes:

A block code is said to be linear **(*n* ,*k*)** code if and only if the 2^k code words from a ***k***- dimensional sub space over a vector space of all ***n***-Tuples over the field ***GF(2)***.

Fields with 2^m symbols are called 'Galois Fields' (pronounced as Galva fields), GF (2^m).Their arithmetic involves binary additions and subtractions. For two valued variables, (***0, 1***).The modulo – 2 addition and multiplication is defined in Fig 6.3.

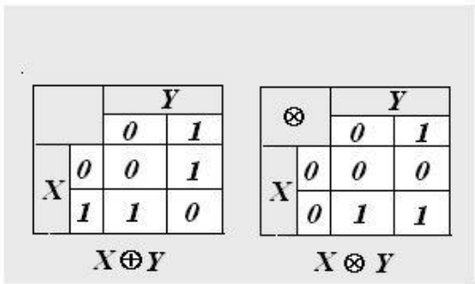


Fig 6.3

⊕	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

⊗	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Fig 6.4

The binary alphabet (***0, 1***) is called a ***field*** of two elements (a binary field and is denoted by ***GF (2)***. (Notice that ⊕ represents the EX-OR operation and ⊗ represents the AND operation).Further in binary arithmetic, $-X=X$ and $X - Y = X \oplus Y$. similarly for 3-valued variables, modulo – 3 arithmetic can be specified as shown in Fig 6.4. However, for brevity while representing polynomials involving binary addition we use + instead of ⊕ and there shall be no confusion about such usage**.**

Polynomials ***f(X)*** with ***1*** or ***0*** as the co-efficients can be manipulated using the above relations. The arithmetic of $GF(2^m)$ can be derived using a polynomial of degree ' ***m***', with binary co-efficients and using a new variable α called the primitive element, such that $p(\alpha) = 0$.When ***p(X)*** is irreducible (i.e. it does not have a factor of degree < ***m*** and >***0***, for example $X^3 + X^2 + 1$, $X^3 + X + 1$, $X^4 + X^3 + 1$, $X^5 + X^2 + 1$ etc. are irreducible polynomials, whereas $f(X)=X^4+X^3+X^2+1$ is not as $f(1) = 0$ and hence has a factor $X+1$) then ***p(X)*** is said to be a ' ***primitive polynomial***'.

If v_n represents a vector space of all ***n***-tuples, then a subset ***S*** of v_n is called a subspace if (i) the all Zero vector is in ***S*** (ii) the sum of any two vectors in ***S*** is also a vector in ***S***. To be more specific, a block code is said to be linear if the following is satisfied. "If v_1 and v_2 are any two code words of length ***n*** of the block code then $v_1 \oplus v_2$ is also a code word length ***n*** of the block code".

Example 6.1: Linear Block code with ***k= 3***, and ***n = 6***

Messages		*Code words*		*Weight (No. of 1's in the code word)*
m_1	000	v_1	0 0 0 0 0 0	0
m_2	001	v_2	0 0 1 1 1 0	3
m_3	010	v_3	0 1 0 1 0 1	3
m_4	100	v_4	1 0 0 0 1 1	3
m_5	011	v_5	0 1 1 0 1 1	4
m_6	101	v_6	1 0 1 1 0 1	4
m_7	110	v_7	1 1 0 1 1 0	4
m_8	111	v_8	1 1 1 0 0 0	3

Observe the linearity property: With v_3 = (***010 101***) and v_4 = (***100 011***), $v_3 \oplus v_4$ = (***110 110***) = v_7.

Remember that ***n*** represents the word length of the code words and ***k*** represents the number of information digits and hence the block code is represented as **(*n*, *k*)** block code.

Thus by definition of a linear block code it follows that if $g_1, g_2 \ldots g_k$ are the k linearly independent code words then every code vector, v, of our code is a combination of these code words, i.e.

$$v = u_1 g_1 \oplus u_2 g_2 \oplus \ldots \oplus u_k g_k \quad \ldots\ldots\ldots\ldots\ldots (6.1)$$

Where $u_j = 0$ or 1, $\forall\ 1 \le j \le k$

Eq (6.1) can be arranged in matrix form by nothing that each g_j is an n-tuple, i.e.

$$g_j = (g_{j1}, g_{j2}, \ldots g_{jn}) \quad \ldots\ldots\ldots\ldots (6.2)$$

Thus we have $v = u\,G$ ..... (6.3)

Where: $u = (u_1, u_2 \ldots u_k)$ ..... (6.4)

represents the data vector and

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{k1} \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix} \quad \ldots\ldots (6.5)$$

is called the "***generator matrix***".

Notice that any k linearly independent code words of an (n, k) linear code can be used to form a Generator matrix for the code. Thus it follows that an (n, k) linear code is completely specified by the k-rows of the generator matrix. Hence the encoder need only to store k rows of G and form linear combination of these rows based on the input message u.

Example 6.2: The $(6, 3)$ linear code of Example 6.1 has the following generator matrix:

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

If $u = m_5$ (say) is the message to be coded, i.e. $u = (011)$

We have $v = u.G = 0.g_1 + 1.g_2 + 1.g_3$

$$= (0,0,0,0,0,0) + (0,1,0,1,0,1) + (0,0,1,1,1,0) = (0, 1, 1, 0, 1, 1)$$

Thus $v = (0\ 1\ 1\ 0\ 1\ 1)$

" v can be computed simply by adding those rows of G which correspond to the locations of 1`s of u."

5.5.2 Systematic Block Codes (Group Property):

A desirable property of linear block codes is the " *Systematic Structure*". Here a code word is divided into two parts –Message part and the redund ant part. If either the first ***k*** digits or the last ***k*** digits of the code word correspond to the message part then we say that the code is a " *Systematic Block Code*". We shall consider systematic codes as depicted i n Fig.6.5.

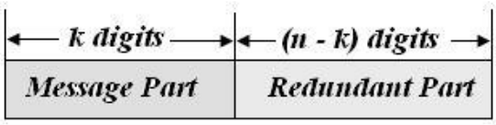


Fig 6.5 Systematic Format of code word.

In the format of Fig.6.5 notice that:

$$v_1 = u_1, v_2 = u_2, v_3 = u_3 \ldots \quad v_k = u_k \qquad \ldots\ldots\ldots\ldots \quad (5.6\ a)$$

$$v_{k+1} = u_1 p_{11} + u_2 p_{21} + u_3 p_{31} + \ldots + u_k p_{k1}$$

$$v_{k+2} = u_1 p_{12} + u_2 p_{22} + u_3 p_{32} + \ldots + u_k p_{k2} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (5.6\ b)$$

M M

$$v_n = u_1 p_{1,n-k} + u_2 p_{2,n-k} + u_3 p_{3,n-k} + \ldots\ldots + u_k p_{k,n-k}$$

Or in matrix from we have

$$[v_1 \quad v_2 \quad \ldots \quad v_k \quad v_{k+1} \quad v_{k+2} \quad \ldots \quad v_n] =$$

$$[u_1 \quad u_2 \quad \ldots \quad u_k] \begin{matrix} 1 & 0 & 0 & \ldots & 0 & p_{11} & p_{12} & \ldots & p_{1,n-k} \\ 0 & 1 & 0 & \ldots & 0 & p_{21} & p_{22} & \ldots & p_{2,n-k} \\ \text{M} & \text{M} & \text{M} & \text{MMM} & \text{M} & \text{M} & \text{M} & \text{MMM} & \text{M} \\ 0 & 0 & 0 & \ldots & 1 & p_{k1} & p_{k2} & \ldots & p_{k,n-k} \end{matrix} \qquad \ldots(5.7)$$

i.e., $v = u.\ G$

Where $G = [I_k, P]$ ..... …. (5.8)

Where $P = \begin{matrix} p_{11} & p_{12} & \ldots & p_{1,n-k} \\ p_{21} & p_{22} & \ldots & p_{2,n-k} \\ \text{M} & \text{M} & & \text{M} \\ p_{k,1} & p_{k,2} & \ldots & p_{k,n-k} \end{matrix}$ ..... (5.9)

I_k is the $k \times k$ identity matrix (unit matrix), P is the $k \times (n-k)$ ' ***parity generator matrix***', in which $p_{i,j}$ are either ***0*** or ***1*** and G is a $k \times n$ matrix. The $(n-k)$ equations given in Eq (6.6*b*) are referred to as ***parity check equations***. Observe that the G matrix of Example 6.2 is in the systematic format. The n-vectors $a = (a_1, a_2 \ldots a_n)$ and $b = (b_1, b_2 \ldots b_n)$ are said to be orthogonal if their inner product defined by:

$$a.b = (a_1, a_2 \ldots a_n)\,(b_1, b_2 \ldots b_n)^T = 0.$$

where, ' ***T'*** represents transposition. Accordingly for any $k \times n$ matrix, G, with k linearly independent rows there exists a $(n-k) \times n$ matrix H with $(n-k)$ linearly independent rows such that any vector in the row space of G is orthogonal to the rows of H and that any vector that is orthogonal to the rows of

H is in the row space of ***G***. Therefore, we can describe an **(*n*, *k*)** linear code generated by ***G*** alternatively as follows:

"An n – tuple, v is a code word generated by G, if and only if $v.H^T = O$***".*** …..(5.9 *a*) (***O* represents an all zero row vector**.)

This matrix ***H*** is called a " ***parity check matrix***" of the code. Its dimension is $(n-k) \times n$.

If the generator matrix has a systematic format, the parity check matrix takes the following form.

$$H = [P^T . I_{n-k}] = \begin{bmatrix} p_{11} & p_{21} & \dots & p_{k1} & 1 & 0 & 0 & \dots & 0 \\ p_{12} & p_{22} & \dots & p_{k2} & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{1,n-k} & p_{2,n-k} & \dots & p_{k,n-k} & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad \text{.....(5.10)}$$

The i^{th} row of ***G*** is:

$$g_i = (0\,0\,\dots 1\,\dots 0\dots 0 \quad p_{i,1} \quad p_{i,2} \dots \quad p_{i,j} \dots \quad p_{i,n-k})$$

(i^{th} element; $(k+j)^{th}$ element)

The j^{th} row of ***H*** is:

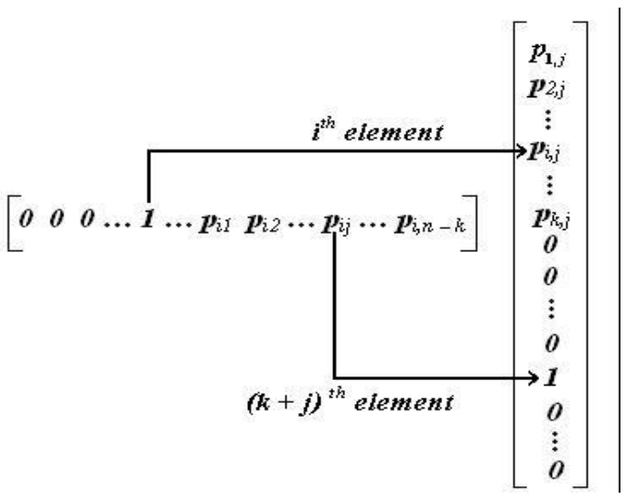
(i^{th} element; $(k+j)^{th}$ element)

$$h_j = (p_{1,j}\, p_{2,j} \dots p_{i,j} \dots p_{k,j}\, 0\,0 \dots 0\,1\,0 \dots 0)$$

Accordingly the inner product of the above ***n*** – vectors is:

$$g_i \times h_j^T = (0\,0\,\dots 1\,\dots 0\dots 0 \quad p_{i,1} \quad p_{i,2} \dots \quad p_{i,j} \dots \quad p_{i,n-k})\,(p_{1,j} \quad p_{2,j} \dots p_{i,j} \dots p_{k,j}\, 0\,0 \dots 0\,1\,0 \dots 0)^T$$

(i^{th} element; $(k+j)^{th}$ element; i^{th} element; $(k+j)^{th}$ element)



$= p_{ij} + p_{ij} = 0$ (as the p_{ij} are either ***0*** or ***1*** and in modulo – 2 arithmetic $X + X = 0$)

This implies simply that:

$$G.H^T = O_{k \times (n-k)} \quad \dots\dots\dots\dots\dots\dots\dots\dots \quad (6.11)$$

Where $\mathbf{O}_{k \times (n-k)}$ is an all zero matrix of dimension $k \times (n-k)$.

Further, since the $(n-k)$ rows of the matrix H are linearly independent, the H matrix of Eq. (6.10) is a parity check matrix of the (n, k) linear systematic code generated by G. Notice that the parity check equations of Eq. (6.6*b*) can also be obtained from the parity check matrix using the fact

$$v.H^T = \mathbf{O}.$$

Alternative Method of proving $v.H^T = \mathbf{O}.$:

We have $v = u.G = u$. $[I_k : P] = [u_1, u_2 \ldots u_k, p_1, p_2 \ldots P_{n-k}]$

Where $p_i = (u_1 p_{1,i} + u_2 p_{2,i} + u_3 p_{3,i} \ldots + u_k p_{k,i})$ are the parity bits found from Eq (6.6*b*).

Now $H' = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix}$

$\therefore v.H^T = [u_1 p_{11} + u_2 p_{21} + \ldots + \ldots + u_k p_{k1} + p_1, u_1 p_{12} + u_2 p_{22} + \ldots + u_k p_{k2} + p_2, \ldots u_1 p_{1,n-k} + u_2 p_{2,n-k} + \ldots + u_k p_{k,n-k} + p_{n-k}]$

$= [p_1 + p_1, p_2 + p_2 \ldots p_{n-k} + p_{n-k}]$

$= [0, 0 \ldots 0]$

Thus $v. H^T = \mathbf{O}.$ This statement implies that an n- Tuple v is a code word generated by G if and only if

$$v H^T = \mathbf{O}$$

Since $v = u G$, This means that: $u G H^T = \mathbf{O}$

If this is to be true for any arbitrary message vector v then this implies: $\mathbf{G H}^T = \mathbf{O}_{k \times (n-k)}$

Example 5.3:

Consider the generator matrix of Example 6.2, the corresponding parity check matrix is

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

5.5.3 Syndrome and Error Detection:

Suppose $v = (v_1, v_2 \ldots v_n)$ be a code word transmitted over a noisy channel and let: $r = (r_1, r_2 \ldots r_n)$ be the received vector. Clearly, r may be different from v owing to the channel noise. The vector sum

$$e = r - v = (e_1, e_2 \ldots e_n) \quad \ldots\ldots\ldots\ldots\ldots \quad (5.12)$$

is an ***n***-tuple, where $e_j = 1$ if $r_j \neq v_j$ and $e_j = 0$ if $r_j = v_j$. This ***n*** – tuple is called the " ***error vector***" or " ***error pattern***". The ***1***'s in ***e*** are the transmission errors caused by the channel noise. Hence from Eq (6.12) it follows:

$$r = v \oplus e \qquad \ldots\ldots\ldots\ldots (5.12a)$$

Observe that the receiver noise does not know either ***v*** or ***e***. Accordingly, on reception of ***r*** the decoder must first identify if there are any transmission errors and, then take action to locate these errors and correct them (***FEC*** – Forward Error Correction) or make a request for re–transmission (***ARQ***). When ***r*** is received, the decoder computes the following **(*n-k*)** tuple:

$$s = r.H^T = (s_1, s_2 \ldots s_{n-k}) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \text{ ..... } (5.13)$$

It then follows from Eq (6.9*a*), that $s = 0$ if and only if ***r*** is a code word and $s \neq 0$ iffy ***r*** is not a code word. This vector s is called " ***The Syndrome***" (a term used in medical science referring to collection of all symptoms characterizing a disease). Thus if $s = 0$, the receiver accepts ***r*** as a valid code word. Notice that there are possibilities of errors undetected, which happens when ***e*** is identical to a nonzero code word. In this case ***r*** is the sum of two code words which according to our linearity property is again a code word. This type of error pattern is referred to an " ***undetectable error pattern***". Since there are $2^k - 1$ nonzero code words, it follows that there are $2^k - 1$ error patterns as well. Hence when an undetectable error pattern occurs the decoder makes a " ***decoding error***".

Eq. (6.13) can be expanded as below:

$$s = r.H^T = (s_1, s_2 \ldots s_{n-k}) = (r_1, r_2, \ldots r_n) \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n-k} \\ p_{21} & p_{22} & \cdots & p_{2,n-k} \\ \vdots & & & \\ p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

From which we have

$$\begin{aligned} s_1 &= r_1 p_{11} + r_2 p_{21} + \ldots + r_k p_{k1} + r_{k+1} \\ s_2 &= r_1 p_{12} + r_2 p_{22} + \ldots + r_k p_{k2} + r_{k+2} \\ &\text{M} \quad \text{M} \quad \text{M} \quad \text{M} \quad \text{M} \\ s_{n-k} &= r_1 p_{1,n-k} + r_2 p_{2,n-k} + \ldots + r_k p_{k,n-k} + r_n \end{aligned} \qquad \ldots(5.14)$$

A careful examination of Eq. (6.14) reveals the following point. The syndrome is simply the vector sum of the received parity digits $(r_{k+1}, r_{k+2} \ldots r_n)$ and the parity check digits recomputed from the received information digits $(r_1, r_2 \ldots r_n)$.

Example 6.4:

We shall compute the syndrome for the (***6, 3***) systematic code of Example 5.2. We have

$$s = (s_1, s_2, s_3) = (r_1, r_2, r_3, r_4, r_5, r_6) \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or $s_1 = r_2 + r_3 + r_4$
$s_2 = r_1 + r_3 + r_5$
$s_3 = r_1 + r_2 + r_6$

In view of Eq. (6.12*a*), and Eq. (6.9*a*) we have

$$s = r.H^T = (v \oplus e)\, H^T$$

$$= v.H^T \oplus e.H^T$$

or $s = e.H^T$ ..... (5.15)

as $v.H^T = O$. Eq. (6.15) indicates that the syndrome depends only on the error pattern and not on the transmitted code word v. For a linear systematic code, then, we have the following relationship between the syndrome digits and the error digits.

$$\begin{aligned} s_1 &= e_1 p_{11} + e_2 p_{21} + \ldots + e_k p_{k,1} + e_{k+1} \\ s_2 &= e_1 p_{12} + e_2 p_{22} + \ldots + e_k p_{k,2} + e_{k+2} \\ &\vdots \\ s_{n-k} &= e_1 p_{1,n-k} + e_2 p_{2,n-k} + \ldots + e_k p_{k,n-k} + e_n \end{aligned} \quad \ldots\ldots\ldots(5.16)$$

Thus, the syndrome digits are linear combinations of error digits. Therefore they must provide us information about the error digits and help us in error correction.

Notice that Eq. (6.16) represents $(n-k)$ linear equations for n error digits – an under-determined set of equations. Accordingly it is not possible to have a unique solution for the set. As the rank of the H matrix is k, it follows that there are 2^k non-trivial solutions. In other words there exist 2^k error patterns that result in the same syndrome. Therefore to determine the true error pattern is not any easy task!

Example 5.5:

For the (*6, 3*) code considered in Example 6 2, the error patterns satisfy the following equations:

$$s_1 = e_2 + e_3 + e_4, \quad s_2 = e_1 + e_3 + e_5, \quad s_3 = e_1 + e_2 + e_6$$

Suppose, the transmitted and received code words are $v = (0\ 1\ 0\ 1\ 0\ 1)$, $r = (0\ 1\ 1\ 1\ 0\ 1)$

Then $s = r.H^T = (1, 1, 0)$

Then it follows that:

$$e_2 + e_3 + e_4 = 1$$
$$e_1 + e_3 + e_5 = 1$$

$$e_1 + e_2 + e_6 = 0$$

There are $2^3 = 8$ error patterns that satisfy the above equations. They are:

$$\{0\,0\,1\,0\,0\,0,\ 1\,0\,0\,0\,0,\ 0\,0\,0\,1\,1\,0,\ 0\,1\,0\,0\,1\,1,\ 1\,0\,0\,1\,0\,1,\ 0\,1\,1\,1\,0\,1,\ 1\,0\,1\,0\,1\,1,\ 1\,1\,1\,1\,1\,0\}$$

To minimize the decoding error, the " ***Most probable error pattern***" that satisfies Eq (6.16) is chosen as the true error vector. For a ***BSC***, the most probable error pattern is the one that has the smallest number of nonzero digits. For the Example 6.5, notice that the error vector ***(0 0 1 0 0 0)*** has the smallest number of nonzero components and hence can be regarded as the most probable error vector. Then using Eq. (6.12) we have

$$\hat{v} = r \oplus e$$

$$= (0\,1\,1\,1\,0\,1) + (0\,0\,1\,0\,0\,0) = (0\,1\,0\,1\,0\,1)$$

Notice now that $\hat{v}$ indeed is the actual transmitted code word.

5.6 Minimum Distance Considerations:

The concept of distance between code words and single error correcting codes was first developed by R .W. Hamming. Let the n-tuples,

$$\alpha = (\alpha_1, \alpha_2 \ldots \alpha_n),\ \beta = (\beta_1, \beta_2 \ldots \beta_n)$$

be two code words. The " ***Hamming distance***" $d(\alpha,\beta)$ between such pair of code vectors is defined as the number of positions in which they differ. Alternatively, using Modulo-2 arithmetic, we have

$$d(\alpha, \beta) \triangleq \sum_{j=1}^{n} (\alpha_j \oplus \beta_j) \qquad \ldots\ldots(5.17)$$

(Notice that Σ represents the usual decimal summation and $\oplus$ is the modulo-2 sum, the EX-OR function).

The " ***Hamming Weight***" $\omega(\alpha)$ of a code vector α is defined as the number of nonzero elements in the code vector. Equivalently, the Hamming weight of a code vector is the distance between the code vector and the ' ***all zero code vector***'.

Example 6.6: Let $\alpha = (0\,1\,1\,1\,0\,1),\ \beta = (1\,0\,1\,0\,1\,1)$

Notice that the two vectors differ in ***4*** positions and hence $d(\alpha,\beta) = 4$. Using Eq (5.17) we find

$$d(\alpha,\beta) = (0 \oplus 1) + (1 \oplus 0) + (1 \oplus 1) + (1 \oplus 0) + (0 \oplus 1) + (1 \oplus 1)$$

$$= 1 + 1 + 0 + 1 + 1 + 0$$

$= 4$ ….. (Here + is the algebraic plus not modulo – 2 sum)

Further, $\omega(\alpha) = 4$ and $\omega(\beta) = 4$.

The " ***Minimum distance***" of a linear block code is defined as the smallest Hamming distance between any pair of code words in the code or the minimum distance is the same as the smallest Hamming weight of the difference between any pair of code words. Since in linear block codes, the sum or difference of two code vectors is also a code vector, it follows then that **" *the minimum distance of a linear block code is the smallest Hamming weight of the nonzero code vectors in the code*".**

The Hamming distance is a metric function that satisfies the triangle inequality. Let α, β and δ be three code vectors of a linear block code. Then

$$d(\alpha,\beta) + d(\beta,\delta) \geq d(\alpha,\delta) \quad \text{.....(5.18)}$$

From the discussions made above, we may write

$$d(\alpha,\beta) = \omega(\alpha \oplus \beta) \quad \text{.....(5.19)}$$

Example 6.7: For the vectors α and β of Example 6.6, we have:

$$\alpha \oplus \beta = (0\oplus1), (1\oplus0), (1\oplus1)\ (1\oplus0), (0\oplus1)\ (1\oplus1) = (11\ 0\ 1\ 1\ 0)$$

$$\therefore\ \omega(\alpha \oplus \beta) = 4 = d(\alpha,\beta)$$

If $\delta = (1\ 0\ 1\ 01\ 0)$, we have $d(\alpha,\beta) = 4;\ d(\beta,\delta) = 1;\ d(\alpha,\delta) = 5$

Notice that the above three distances satisfy the triangle inequality:

$$d(\alpha,\beta) + d(\beta,\delta) = 5 = d(\alpha,\delta)$$

$$d(\beta,\delta) + d(\alpha,\delta) = 6 > d(\alpha,\beta)$$

$$d(\alpha,\delta) + d(\alpha,\beta) = 9 > d(\beta,\delta)$$

Similarly, the minimum distance of a linear block code, ' ***C***" may be mathematically represented as below:

$$d_{min} = Min\ \{d(\alpha,\beta): \alpha,\beta \in C,\ \alpha \neq \beta\} \quad \text{.....(5.20)}$$

$$= Min\ \{\omega(\alpha \oplus \beta): \alpha,\beta \in C,\ \alpha \neq \beta\}$$

$$= Min\ \{\omega(v),\ v \in C,\ v \neq 0\} \quad \text{..... (5 .21)}$$

That is $d_{min} \triangleq \omega_{min}$. The parameter ω_{min} is called the " ***minimum weight***" of the linear code ***C***.The minimum distance of a code, d_{min}, is related to the parity check matrix, ***H***, of the code in a fundamental way. Suppose ***v*** is a code word. Then from Eq. (6.9*a*) we have:

$$0 = v.H^T$$

$$= v_1h_1 \oplus v_2h_2 \oplus \ldots \oplus v_nh_n$$

Here $h_1, h_2 \ldots h_n$ represent the columns of the ***H*** matrix. Let $v_{j1}, v_{j2} \ldots v_{jl}$ be the ' ***l***' nonzero components of ***v*** i.e. $v_{j1} = v_{j2} = \ldots v_{jl} = 1$. Then it follows that:

$$h_{j1} \oplus h_{j2} \oplus \ldots \oplus h_{jl} = O^T \qquad \ldots\ldots\ldots (5.22)$$

That is " ***if v is a code vector of Hamming weight 'l', then th ere exist 'l' columns of H such that the vector sum of these columns is equal to the zero vector***". Suppose we form a binary ***n***-tuple of weight ' ***l***', viz. $x = (x_1, x_2 \ldots x_n)$ whose nonzero components are $x_{j1}, x_{j2} \ldots x_{jl}$. Consider the product:

$$x.H^T = x_1h_1 \oplus x_2h_2 \oplus \ldots. \oplus x_nh_n = x_{j1}h_{j1} \oplus x_{j2}h_{j2} \oplus \ldots. \oplus x_{jl}h_{jl} = h_{j1} \oplus h_{j2} \oplus \ldots \oplus h_{jl}$$

If Eq. (6.22) holds, it follows $x.H^T = O$ and hence ***x*** is a code vector. Therefore, we conclude that ***"if there are 'l' columns of H matrix whose vector sum is the zero vector then there exists a code vector of Hamming weight 'l' ".***

From the above discussions, it follows that:

i) If no ***(d-1)*** or fewer columns of ***H*** add to O^T, the all zero column vector, the code has a minimum weight of at least' ***d***'.

ii) The minimum weight (or the minimum distance) of a linear block code ***C***, is the smallest number of columns of ***H*** that sum to the all zero column vector.

For the H matrix of Example 6.3, i.e. $H = \begin{matrix} 0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0 \\ 1\,1\,0\,0\,0\,1 \end{matrix}$, notice that all columns of ***H*** are non zero and distinct. Hence no two or fewer columns sum to zero vector. Hence the minimum weight of the code is at least 3.Further notice that the 1^{st}, 2^{nd} and 3^{rd} columns sum to O^T. Thus the minimum weight of the code is ***3***. We see that the minimum weight of the code is indeed ***3*** from the table of Example 6.1.

5.6.1 Error Detecting and Error Correcting Capabilities:

The minimum distance, d_{min}, of a linear block code is an important parameter of the code. To be more specific, it is the one that determines the error correcting capability of the code. To understand this we shall consider a simple example. Suppose we consider ***3***-bit code words plotted at the vertices of the cube as shown in Fig.6.10.

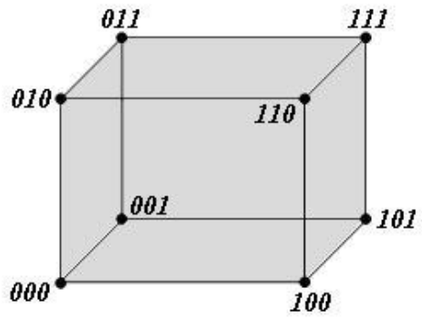


Fig 6.10 The Distance concept

Clearly, if the code words used are {***000, 101, 110, 011***}, the Hamming distance between the words is ***2***. Notice that any error in the received words locates them on the vertices of the cube which are not code words and may be recognized as single errors. The code word pairs with Hamming distance = ***3*** are: **(*000, 111*), (*100, 011*), (*101, 010*)** and **(*001, 110*)**. If a code word **(*000*)** is received as **(*100, 010, 001*)**, observe that these are nearer to **(*000*)** than to **(*111*)**. Hence the decision is made that the transmitted word is **(*000*)**.

Suppose an **(*n, k*)** linear block code is required to detect and correct all error patterns (over a ***BSC***), whose Hamming weight, $\omega \le t$. That is, if we transmit a code vector α and the received vector is $\beta = \alpha \oplus e$, we want the decoder out put to be $\hat{\alpha} = \alpha$ subject to the condition $\omega(e) \le t$.

Further, assume that 2^k code vectors are transmitted with equal probability. The best decision for the decoder then is to pick the code vector nearest to the received vector β for which the Hamming distance is the smallest. i.e., $d(\alpha,\beta)$ is minimum. With such a strategy the decoder will be able to detect and correct all error patterns of Hamming weight $\omega(e) \le t$ provided that the minimum distance of the code is such that:

$$d_{min} \ge (2t + 1) \qquad \text{.....(5.23)}$$

d_{min} is either odd or even. Let ' t' be a positive integer such that

$$2t + 1 \le d_{min} \le 2t + 2 \qquad \text{..... (5.24)}$$

Suppose δ be any other code word of the code. Then, the Hamming distances among α,β and δ satisfy the triangular inequality:

$$d(\alpha,\beta) + d(\beta,\delta) \ge d(\alpha,\delta) \qquad \text{..... (5.25)}$$

Suppose an error pattern of ' t'' errors occurs during transmission of α. Then the received vector β differs from α in ' t'' places and hence $d(\alpha,\beta) = t'$. Since α and δ are code vectors, it follows from Eq. (6.24).

$$d(\alpha,\delta) \ge d_{min} \ge 2t + 1 \qquad \text{.....(5.26)}$$

Combining Eq. (6.25) and (6.26) and with the fact that $d(\alpha,\beta) = t'$, it follows that:

$$d(\delta, \beta) \ge 2t + 1 - t' \qquad \text{.....(5.27)}$$

Hence if $t' \le t$, then: $d(\delta,\beta) > t$ .....(528)

Eq 6.28 says that if an error pattern of ' t' or fewer errors occurs, the received vector β is closer (in Hamming distance) to the transmitted code vector α than to any other code vector δ of the code. For a BSC, this means $P(\beta|\alpha) > P(\beta|\delta)$ for $\alpha \ne \delta$. Thus based on the maximum likelihood decoding scheme, β is decoded as α , which indeed is the actual transmitted code word and this results in the correct decoding and thus the errors are corrected.

On the contrary, the code is not capable of correcting error patterns of weight $l > t$. To show this we proceed as below:

Suppose $d(\alpha,\delta) = d_{min}$, and let e_1 and e_2 be two error patterns such that:

i) $e_1 \oplus e_2 = \alpha \oplus \delta$

ii) e_1 and e_2 do not have nonzero components in common places. Clearly,

$$\omega(e_1) + \omega(e_2) = \omega(\alpha \oplus \delta) = d(\alpha, \delta) = d_{min} \quad \text{.....(5.29)}$$

Suppose, α is the transmitted code vector and is corrupted by the error pattern e_1. Then the received vector is:

$$\beta = \alpha \oplus e_1 \quad \text{..... (5.30)}$$

and $d(\alpha,\beta) = \omega(\alpha \oplus \beta) = \omega(e_1)$ ..... (5.31)

$$d(\delta,\beta) = \omega(\delta \oplus \beta)$$

$$= \omega(\delta \oplus \alpha \oplus e_1) = \omega(e_2) \quad \text{.....(5.32)}$$

If the error pattern e_1 contains more than' t' errors, i.e. $\omega(e_1) > t$, and since $2t + 1 \leq d_{min} \leq 2t + 2$, it follows

$$\omega(e_2) \leq t - 1 \quad \text{.....(5.33)}$$

$$\therefore d(\alpha,\beta) \geq d(\delta,\beta) \quad \text{..... (5.34)}$$

This inequality says that there exists an error pattern of $l > t$ errors which results in a received vector closer to an incorrect code vector i.e. based on the maximum likelihood decoding scheme decoding error will be committed.

To make the point clear, we shall give yet another illustration. The code vectors and the received vectors may be represented as points in an n- dimensional space. Suppose we construct two spheres, each of equal radii,' t' around the points that represent the code vectors α and δ. Further let these two spheres be mutually exclusive or disjoint as shown in Fig.6.11 (*a*).

For this condition to be satisfied, we then require $d(\alpha,\delta) \geq 2t + 1$.In such a case if $d(\alpha,\beta) \leq t$, it is clear that the decoder will pick α as the transmitted vector.

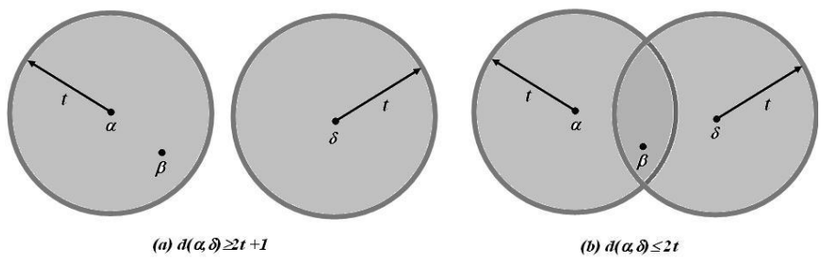


Fig 6.11 Illustrating the distance concept

On the other hand, if $d(\alpha,\delta) \leq 2t$, the two spheres around α and δ intersect and if 'β' is located as in Fig. 6.11(*b*), and α is the transmitted code vector it follows that even if $d(\alpha,\beta) \leq t$, yet β is as close to δ as it is to α. The decoder can now pick δ as the transmitted vector which is wrong. Thus it is imminent that "an (n, k) linear block code has the power to correct all error patterns of weight' t' or less if and only if $d(\alpha,\delta) \geq 2t + 1$ for all α and δ". However, since the smallest distance between any pair of code words is the minimum distance of the code, d_{min} , 'guarantees' correcting all the error patterns of

$$t \leq \frac{1}{2}(d_{min} - 1) \quad \ldots\ldots\ldots (5.35)$$

where $\frac{1}{2}(d_{min} - 1)$ denotes the largest integer no greater than the number $\frac{1}{2}(d_{min} - 1)$. The parameter 't' = $\frac{1}{2}(d_{min} - 1)$ is called the "***random-error-correcting capability***" of the code and the code is referred to as a "***t-error correcting code***". The (***6, 3***) code of Example 6.1 has a minimum distance of ***3*** and from Eq. (6.35) it follows ***t = 1***, which means it is a '***Single Error Correcting'*** (***SEC***) code. It is capable of correcting any error pattern of single errors over a block of six digits.

For an ***(n, k)*** linear code, observe that, there are 2^{n-k} syndromes including the all zero syndrome. Each syndrome corresponds to a specific error pattern. If 'j' is the number of error locations in the ***n***-dimensional error pattern ***e***, we find in general, there are $\binom{n}{j} = nC_j$ multiple error patterns. It then follows that the total number of all possible error patterns = $\sum_{j=0}^{t} \binom{n}{j}$, where '$t$' is the maximum number of error locations in ***e***. Thus we arrive at an important conclusion. *"* ***If an (n, k) linear block code is to be capable of correcting up to't' errors, the total number of syndromes shall not be less than the total number of all possible error patterns****",* i.e**.**

$$2^{n-k} \geq \sum_{j=0}^{t} \binom{n}{j} \quad \ldots\ldots(5.36)$$

Eq (6.36) is usually referred to as the "***Hamming bound***". A binary code for which the Hamming Bound turns out to be equality is called a "***Perfect code***".

6.7 Standard Array and Syndrome Decoding:

The decoding strategy we are going to discuss is based on an important property of the syndrome.

Suppose v_j , $j = 1, 2 \ldots 2^k$, be the 2^k distinct code vectors of an ***(n, k)*** linear block code. Correspondingly let, for any error pattern ***e***, the 2^k distinct error vectors, e_j, be defined by

$$e_j = e \oplus v_j \text{ , } j = 1, 2 \ldots 2^k \quad \ldots\ldots\ldots\ldots\ldots\ldots (5.37)$$

The set of vectors $\{e_j, j = 1, 2 \ldots 2^k\}$ so defined is called the "***co- set***" of the code. That is, a '***co-set***' contains exactly 2^k elements that differ at most by a code vector. It then fallows that there are 2^{n-k} ***co- sets*** for an ***(n, k)*** linear block code. Post multiplying Eq (6.37) by H^T, we find

$$e_j H^T = eH^T \oplus v_j H^T$$
$$= e H^T \quad \ldots\ldots\ldots\ldots(5.38)$$

Notice that the RHS of Eq (6.38) is independent of the index ***j***, as for any code word the term $v_j H^T = 0$. From Eq (6.38) it is clear that "***all error patterns that differ at most by a code word have the same syndrome***". That is, each co-set is characterized by a uniqu e syndrome.

Since the received vector r may be any of the 2^n n-tuples, no matter what the transmitted code word was, observe that we can use Eq (6.38) to partition the received code words into 2^k disjoint sets and try to identify the received vector. This will be done by preparing what is called the " ***standard array***". The steps involved are as below:

Step1: Place the 2^k code vectors of the code in a row, with the all zero vector $v_1 = (0, 0, 0 \ldots 0) = O$ as the first (left most) element.

Step 2: From among the remaining $(2^n - 2^k)$ - n – tuples, e_2 is chosen and placed below the all-zero vector, v_1. The second row can now be formed by placing $(e_2 \oplus v_j)$, $j = 2, 3 \ldots 2^k$ under v_j

Step 3: Now take an un-used n-tuple e_3 and complete the 3^{rd} row as in ***step 2***.

Step 4: continue the process until all the n-tuples are used.

The resultant array is shown in Fig. 5.12.

$v_1 = O$	v_2	v_3	……	v_2k
e_2	$v_2 \oplus e_2$	$v_3 \oplus e_2$	……	$v_2^k \oplus e_2$
e_3	$v_2 \oplus e_3$	$v_3 \oplus e_3$	……..	$v_2^k \oplus e_3$
⋮	⋮	⋮	⋮	⋮
e_2^{n-k}	$v_2 \oplus e_2^{n-k}$	$v_3 \oplus e_2^{n-k}$	…….	$v_2^k \oplus e_2^{n-k}$

Fig 6.12 Standard Array for an (n, k) linear Block code.

Since all the code vectors, v_j, are all distinct, the vectors in any row of the array are also distinct. For, if two n-tuples in the l-th row are identical, say $e_l \oplus v_j = e_l \oplus v_m$, $j \neq m$; we should have $v_j = v_m$ which is impossible. Thus it follows that ***"no two n-tuples in the same row of a standard array are identical".***

Next, let us consider that an n-tuple appears in both l-th row and the m-th row. Then for some j_1 and j_2 this implies $e_l \oplus v_{j1} = e_m \oplus v_{j2}$, which then implies $e_l = e_m \oplus (v_{j2} \oplus v_{j1})$; (remember that $X \oplus X = 0$ in modulo-2 arithmetic) or $e_l = e_m \oplus v_{j3}$ for some j_3. Since by property of linear block codes v_{j3} is also a code word, this implies, by the construction rules given, that e_l must appear in the m-th row, which is a contradiction of our steps, as the first element of the m-th row is e_m and is an unused vector in the previous rows. This clearly demonstrates another important property of the array: ***"Every n-tuple appears in one and only one row".***

From the above discussions it is clear that there are 2^{n-k} disjoint rows or co-sets in the standard array and each row or co-set consists of 2^k distinct entries. The first n-tuple of each co-set, (i.e., the entry in the first column) is called the " ***Co-set leader***". Notice that any element of the co-set can be used as a co-set leader and this does not change the element of the co-set - it results simply in a permutation.

Suppose D_j^T is the j^{th} column of the standard array. Then it follows

$$D_j = \{v_j, e_2 \oplus v_j, e_3 \oplus v_j \ldots e_2^{n-k} \oplus v_j\} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6.39)$$

where v_j is a code vector and $e_2, e_3, \ldots e_{2^{n-k}}$ are the co-set leaders.

The 2^k disjoints columns $D_1^T, D_2^T \ldots D_{2^k}^T$ can now be used for decoding of the code. If v_j is the transmitted code word over a noisy channel, it follows from Eq (6.39) that the received vector r is in D_j^T if the error pattern caused by the channel is a co-set leader. If this is the case r will be decoded correctly as v_j. If not an erroneous decoding will result for, any error pattern $\hat{e}$ which is not a co-set leader must be in some co-set and under some nonzero code vector, say, in the i-th co-set and under $v \neq 0$. Then it follows

$$\hat{e} = e_i \oplus v_l \text{ , and the received vector is } r$$

$$= v_j \oplus e = v_j \oplus (e_i \oplus v_l) = e_i \oplus v_m$$

Thus the received vector is in D_m^T and it will be decoded as v_m and a decoding error has been committed. Hence it is explicitly clear that ***"Correct decoding is possible if and only if the error pattern caused by the channel is a co-set leader"*** . Accordingly, the 2^{n-k} co-set leaders (including the all zero vector) are called the " ***Correctable error patterns***", and it follows " ***Every (n, k) linear block code is capable of correcting 2^{n-k} error patterns***".

So, from the above discussion, it follows that in order to minimize the probability of a decoding error, " ***The most likely to occur***" error patterns should be chosen as co-set leaders . For a ***BSC*** an error pattern of smallest weight is more probable than that of a larger weight. Accordingly, when forming a standard array, error patterns of smallest weight should be chosen as co-set leaders. Then the decoding based on the standard array would be the ' ***minimum distance decoding***' (the maximum likelihood decoding). This can be demonstrated as below.

Suppose a received vector r is found in the j^{th} column and l^{th} row of the array. Then r will be decoded as v_j. We have

$$d(r, v_j) = \omega(r \oplus v_j) = \omega(e_l \oplus v_j \oplus v_j) = \omega(e_l)$$

where we have assumed v_j indeed is the transmitted code word. Let v_s be any other code word, other than v_j. Then

$$d(r, v_s) = \omega(r \oplus v_s) = \omega(e_l \oplus v_j \oplus v_s) = \omega(e_l \oplus v_i)$$

as v_j and v_s are code words, $v_i = v_j \oplus v_s$ is also a code word of the code. Since e_l and $(e_l \oplus v_i)$ are in the same co set and, that e_l has been chosen as the co-set leader and has the smallest weight it follows $\omega(e_l) \leq \omega(e_l \oplus v_i)$ and hence $d(r, v_j) \leq d(r, v_s)$. Thus the received vector is decoded into a closet code vector. Hence, if each co-set leader is chosen to have minimum weight in its co-set, the standard array decoding results in the minimum distance decoding or maximum likely hood decoding.

Suppose ***"$a_0, a_1, a_2 \ldots, a_n$"*** denote the number of co-set leaders with weights ***0, 1, 2… n.*** This set of numbers is called the " ***Weight distribution***" of the co-set leaders. Since a decoding error wil l occur if and only if the error pattern is not a co-set leader, the probability of a decoding error for a BSC with error probability (transition probability) p is given by

$$P(E) = 1 - \sum_{j=0}^{n} a_j p^j (1-p)^{n-j} \qquad \ldots\ldots\ldots\ldots\ldots(5.40)$$

Example 6.8:

For the (6, 3) linear block code of Example 6.1 the standard array, along with the syndrome table, is as below:

Standard Array for the (6,3) linear block code of Example 6.1

Syndrome	*Co-set Leader*							
000	*000 000*	*001 110*	*010 101*	*011 011*	*100 011*	*101 101*	*110 110*	*111 000*
001	*000 001*	*001 111*	*010 100*	*011 010*	*100 010*	*101 100*	*110 111*	*111 001*
010	*000 010*	*001 100*	*010 111*	*011 001*	*100 001*	*101 111*	*110 100*	*111010*
100	*000 100*	*001 010*	*010 001*	*011 111*	*100 111*	*101 001*	*110 010*	*111 100*
110	*001 000*	*000 110*	*011 101*	*010 011*	*101 011*	*100 101*	*111 110*	*110 000*
101	*010 000*	*011 110*	*000 101*	*001 011*	*110 011*	*111 101*	*100 110*	*101 000*
011	*100 000*	*101 110*	*110 101*	*111 011*	*000 011*	*001 101*	*010 110*	*011 000*
111	*001 001*	*000 111*	*011 100*	*010 010*	*101 010*	*100 100*	*111 111*	*110 001*

The weight distribution of the co-set leaders in the array shown are $a_0 = 1, a_1 = 6, a_2 = 1, a_3 = a_4 = a_5 = a_6 = 0$.From Eq (6.40) it then follows:

$$P(E) = 1 - [(1-p)^6 + 6p(1-p)^5 + p^2(1-p)^4]$$

$$\text{With } p = 10^{-2}, \text{ we have } P(E) = 1.3643879 \times 10^{-3}$$

A received vector ***(010 001)*** will be decoded as ***(010101)*** and a received vector ***(100 110)*** will be decoded as ***(110 110).***

We have seen in Eq. (6.38) that each co-set is characterized by a unique syndrome or there is a one- one correspondence between a co- set leader (a correctable error pattern) and a syndrome. These relationships, then, can be used in preparing a decoding table that is made up of 2^{n-k} co-set leaders and their corresponding syndromes. This table is either stored or wired in the receiver. The following are the steps in decoding:

Step 1: Compute the syndrome $s = r.H^T$

Step 2: Locate the co-set leader e_j whose syndrome is s. Then e_j is assumed to be the error pattern caused by the channel.

Step 3: Decode the received vector r into the code vector $v = r \oplus e_j$

This decoding scheme is called the " ***Syndrome decoding***" or the " ***Table look up decoding***".

Observe that this decoding scheme is applicable to any linear (***n, k***) code, i.e., it need not necessarily be a systematic code.

Comments:

1) Notice that for all correctable single error patterns the syndrome will be identical to a column of the ***H*** matrix and indicates that the received vector is in error corresponding to that column position.

For Example, if the received vector is (***010001***), then the syndrome is (***100***). This is identical withthe 4^{th} column of the ***H***- matrix and hence the 4^{th} – position of the received vector is in error. Hence the corrected vector is ***010101***. Similarly, for a received vector (***100110***), the syndrome is ***101***

and this is identical with the second column of the ***H***-matrix. Thus the second position of the received vector is in error and the corrected vector is (***110110***).

2) A table can be prepared relating the error locations and the syndrome. By suitable combinatorial circuits data recovery can be achieved. For the (***6, 3***) systematic linear code we have the following table for $r = (r_1\ r_2\ r_3\ r_4\ r_5\ r_6)$.

Error location	*Error in digits*	*Syndrome*
1	r_1	*0 1 1*
2	r_2	*1 0 1*
3	r_3	*1 1 0*
4	r_4	*1 0 0*
5	r_5	*0 1 0*
6	r_6	*0 0 1*
No error		*0 0 0*

5.8 Hamming Codes:

Hamming code is the first class of linear block codes devised for error correction. The single error correcting (***SEC***) Hamming codes are characterized by the following parameters.

Code length: $n = (2^m - 1)$

Number of Information symbols: $k = (2^m - m - 1)$

Number of parity check symbols :$(n - k) = m$

Error correcting capability: $t = 1, (d_{min} = 3)$

The parity check matrix ***H*** of this code consists of all the non-zero ***m***-tuples as its columns. In systematic form, the columns of ***H*** are arranged as follows

$H = [Q \mid I_m]$

Where I_m is an identity (unit) matrix of order $m \times m$ and ***Q*** matrix consists of

$(2^m - m - 1)$ columns which are the ***m***-tuples of weight ***2*** or more. As an illustration for ***k=4*** we have from $k = 2^m - m - 1$.

m=1 k=0, m=2 k=1, m=3 k=4

Thus we require ***3*** parity check symbols and the length of the code $2^3 - 1 = 7$. This results in the ***(7, 4)*** Hamming code.

The parity check matrix for the (***7, 4***) linear systematic Hamming code is then

$$H = \left[\begin{array}{c:c} 1110 & 100 \\ 1101 & 010 \\ 1011 & 001 \end{array}\right]$$

The generator matrix of the code can be written in the form

$$G = \left[I_{2^m - m - 1} \; M \; Q^T \right]$$

And for the ***(7, 4)*** systematic code it follows:

$$G = \left[\begin{array}{c:c} 1000 & 111 \\ 0100 & 110 \\ 0010 & 101 \\ 0001 & 011 \end{array}\right]$$

$p_1 \; p_2 \; m_1 \; p_3 \; m_2 \; m_3 \; m_4 \; p_4 \; m_5 \; m_6 \; m_7 \; m_8 \; m_9 \; m_{10} \; m_{11} \; p_5 \; m_{12}$

Where $p_1, p_2, p_3 \ldots$ are the parity digits and $m_1, m_2, m_3 \ldots$ are the message digits. For example, let us consider the non systematic ***(7, 4)*** Hamming code.

$p_1 = 1, 3, 5, 7, 9, 11, 13, 15 \ldots$

$p_2 = 2, 3, 6, 7, 10, 11, 14, 15 \ldots$

$p_3 = 4, 5, 6, 7, 12, 13, 14, 15 \ldots$

It can be verified that ***(7, 4), (15, 11), (31, 26), (63, 57)*** are all single error correcting Hamming codes and are regarded quite useful.

An important property of the Hamming codes is that they satisfy the condition of Eq. (6.36) with equality sign, assuming that $t=1$.This means that Hamming codes are " ***single error correcting binary perfect codes***". This can also be verified from Eq. (6.35)

We may delete any ' l 'columns from the parity check matrix H of the Hamming code resulting in the reduction of the dimension of H matrix to $m \times (2^m - l - 1)$.Using this new matrix as the parity check matrix we obtain a " ***shortened***" Hamming code with the following parameters.

Code length: $n = 2^m - l - 1$

Number of Information symbols: $k = 2^m - m - l - 1$

Number of parity check symbols: $n - k = m$

Minimum distance: $d_{min} \geq 3$

Notice that if the deletion of the columns of the H matrix is proper, we may obtain a Hamming code with $d_{min} = 4$.For example if we delete from the sub-matrix Q all the columns of even weight, we obtain an $m \times 2^{m-1}$ matrix

$$\overline{H} = \left[\overline{Q} \; : I_m \right]$$

Where $\overline{Q}$ contains $(2^{m-1} - m)$ columns of odd weight. Clearly no three columns add to zero as all columns have odd weight .However, for a column in $\overline{Q}$, there exist three columns in I_m such that four columns add to zero .Thus the shortened Hamming codes with $\overline{H}$ as the parity check matrix has minimum distance exactly ***4***.

The distance – ***4*** shortened Hamming codes can be used for correcting all single error patterns while simultaneously detecting all double error patterns. Notice that when single errors occur the syndromes contain odd number of one's and for double errors it contains even number of ones. Accordingly the decoding can be accomplished in the following manner.

(1) If $s = 0$, no error occurred.

(2) If ***s*** contains odd number of ones, single error has occurred .The single error pattern pertaining to this syndrome is added to the received code vector for error correction.

(3) If ***s*** contains even number of one's an uncorrectable error pattern has been detected.

Alternatively the ***SEC*** Hamming codes may be made to detect double errors by adding an extra parity check in its $(n+1)^{Th}$ position. Thus (***8, 4***), (***6, 11***) etc. codes have $d_{min} = 4$ and correct single errors with detection of double errors.

Review Questions:

1. Design a single error correcting code with a message block size of 11 and show that by an example that it can correct single error.

2. If C_i and C_j an two code vectors in a (n,k) linear block code, show that their sum is also a code vector.

3. Show $CH^T=0$ for a linear block code.

4. Prove that the minimum distance of a linear block code is the smallest weight of the non-zero code vector in the code.

5. What is error control coding? Which are the functional blocks of a communication system that accomplish this? Indicate the function of each block. What is the error detection and correction on the performance of communication system?

6. Explain briefly the following:
   a. Golay code
   b. BCH Code

7. Explain the methods of controlling errors

8. List out the properties of linear codes.

9. Explain the importance of hamming codes & how these can be used for error detection and correction.

10. Write a standard array for (7.4) code

UNIT – 6

Syllabus:

Binary Cycle Codes, Algebraic structures of cyclic codes, Encoding using an (n- k) bit shift register, Syndrome calculation. BCH codes. **7 Hours**

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996. Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007

Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

UNIT – 6

We are, in general, not very much concerned in our every daily life with accurate transmission of information. This is because of the redundancy associated with our language-in conversations, lectures, and radio or telephone communications. Many words or even sentences may be missed still not distorting the meaning of the message.

However, when we are to transmit intelligence-more information in a shorter time, we wish to eliminate unnecessary redundancy. Our language becomes less redundant and errors in transmission become more serious. Notice that while we are talking about numerical data, misreading of even a single digit could have a marked effect on the intent of the message. Thus the primary objective of coding for transmission of intelligence would be two fold – ***increase*** the efficiency and ***reduce*** the transmission errors. Added to this we would like our technique to ensure security and reliability. In this chapter we present some techniques for source encoding and connection between coding and information theory in the light of Shannon's investigation. The problem of channel encoding- coding for error detection and correction will be taken up in the next chapter.

6.1 Definition of Codes:

'Encoding' or 'Enciphering' is a procedure for asso ciating words constructed from a finite alphabet of a language with given words of another language in a one-to- one manner.

Let the source be characterized by the set of symbols

$$S= \{s_1, s_2 \ldots s_q\} \quad \ldots\ldots\ldots \quad (6.1)$$

We shall call ' ***S***' as the " ***Source alphabet***". Consider another set, ***X***, comprising of ' ***r*** ' symbols.

$$X=\{x_1, x_2 \ldots x_r\} \quad \ldots\ldots\ldots \quad (6.2)$$

We shall call ' ***X***' as the " ***code alphabet***". We define " ***coding***" as the ***mapping*** of all possible sequences of symbols of ***S*** into sequences of symbol of ***X***. In other words " ***coding means representing each and every symbol of S by a sequence of symbols of X such that there shall be a one-to-one relationship"*** Any finite sequence of symbols from an alphabet will be called a " ***Word***". Thus any sequence from the alphabet ' ***X***' forms a " ***code word***". The total number of symbols contained in the ' ***word***' will be called " ***word length***". For example the sequences { x_1 ; $x_1x_3x_4$; $x_3x_5x_7x_9$; $x_1x_1x_2x_2x_2$} *form code words. Their word lengths are respectively* ***1; 3; 4****; and* ***5****.The* sequences {***100001001100011000***} and {***110011110000111100011100***} are binary code words with word lengths ***18*** and ***25*** respectively.

6.2 Basic properties of codes:

The definition of codes given above is very broad and includes many undesirable properties. In order that the definition is useful in code synthesis, we require the codes to satisfy certain properties. We shall intentionally take trivial examples in order to get a better understanding of the desired properties.

1. Block codes:

A block code is one in which a particular message of the source is always encoded into the same " ***fixed sequence***" of the code symbol. Although, in general, block m eans ' *a group having identical property*' we shall use the word here to mean a ' *fixed sequence*' only. Accordingly, the code can be a ' ***fixed length code***' or a " ***variable length code***" and we shall be concentrating on the latter type in this chapter. To be more specific as to what we mean by a block code, consider a communication system with one transmitter and one receiver. Information is transmitted using certain set of code words. If the transmitter wants to change the code set, first thing to be done is to inform the receiver. Other wise the receiver will never be able to understand what is being transmitted. Thus, until and unless the receiver is informed about the changes made you are not permitted to change the code set. In this sense the code words we are seeking shall be always ***finite sequences*** of the code alphabet-they are ***fixed sequence codes***.

Example 6.1: Source alphabet is $S = \{s_1, s_2, s_3, s_4\}$, Code alphabet is $X = \{0, 1\}$ and The Code words are: $C = \{0, 11, 10, 11\}$

2. Non – singular codes:

A block code is said to be non singular if all the words of the code set X_1, are "distinct". The codes given in Example 6.1 do not satisfy this property as the codes for s_2 and s_4 are not different. We can not distinguish the code words. If the codes are not distinguishable on a simple inspection we say the code set is " ***singular in the small***". We modify the code as below.

Example 6.2: $S = \{s_1, s_2, s_3, s_4\}$, $X = \{0, 1\}$; Codes, $C = \{0, 11, 10, 01\}$

However, the codes given in Example 6.2 although appear to be non-singular, upon transmission would pose problems in decoding. For, if the transmitted sequence is ***0011***, it might be interpreted as $s_1 s_1 s_4$ or $s_2 s_4$. Thus there is an ambiguity about the code. No doubt, the code is non-singular in the small, but becomes *" **Singular in the large**"*.

3. Uniquely decodable codes:

A non-singular code is uniquely decipherable, if every word immersed in a sequence of words can be uniquely identified. The n^{th} extension of a code, that maps each message into the code words C, is defined as a code which maps the sequence of messages into a sequence of code words. This is also a block code, as illustrated in the following example.

Example 6.3: Second extension of the code set given in Example 6.2.

$S^2 = \{s_1s_1, s_1s_2, s_1s_3, s_1s_4; s_2s_1, s_2s_2, s_2s_3, s_2s_4, s_3s_1, s_3s_2, s_3s_3, s_3s_4, s_4s_1, s_4s_2, s_4s_3, s_4s_4\}$

Source Symbols	Codes	Source Symbols	Codes	Source Symbols	Codes	Source Symbols	Codes
s_1s_1	0 0	s_2s_1	1 1 0	s_3s_1	1 0 0	s_4s_1	0 1 0
s_1s_2	0 1 1	s_2s_2	1 1 1 1	s_3s_2	1 0 1 1	s_4s_2	0 1 1 1
s_1s_3	0 1 0	s_2s_3	1 1 1 0	s_3s_3	1 0 1 0	s_4s_3	0 1 1 0
s_1s_4	0 0 1	s_2s_4	1 1 0 1	s_3s_4	1 0 0 1	s_4s_4	0 1 0 1

Notice that, in the above example, the codes for the source sequences, s_1s_3 and s_4s_1 are not distinct and hence the code is *" **Singular in the Large**"*. Since such singularity properties introduce

ambiguity in the decoding stage, we therefore require, in general, for unique decidability of our codes that **" *The n^{th} extension of the code be non-singular for every finite n.*"**

4. Instantaneous Codes:

A uniquely decodable code is said to be " ***instantaneous***" if the end of any code word is recognizable with out the need of inspection of succeeding code symbols. That is ***there is no time lag in the process of decoding***. To understand the concept, consider the following codes:

Example 6.4:

Source symbols	*Code A*	*Code B*	*Code C*
s_1	0 0	0	0
s_2	0 1	1 0	0 1
s_3	1 0	1 1 0	0 1 1
s_4	1 1	1 1 1 0	0 1 1 1

Code A undoubtedly is the simplest possible uniquely decipherable code. It is non- singular and all the code words have same length. The decoding can be done as soon as we receive two code symbols without any need to receive succeeding code symbols.

Code B is also uniquely decodable with a special feature that the ***0***`s indicate the termination of a code word. It is called the " ***comma code***". When scanning a sequence of code symbols, we may use the comma to determine the end of a code word and the beginning of the other. Accordingly, notice that the codes can be decoded as and when they are received and there is, once again, no time lag in the decoding process.

Where as, although ***Code C*** is a non- singular and uniquely decodable code it cannot be decoded word by word as it is received. For example, if we receive ' ***01***', we cannot decode it as ' s_2' until we receive the next code symbol. If the next code symbol is ' ***0***',indeed the previous word corresponds to s_2, while if it is a ' ***1***' it may be the symbol s_3; which can be concluded so if only if we receive a ' ***0***'in the fourth place. Thus, there is a definite ' ***time lag***' before a word can be decoded. Such a ' *time waste*' is not there if we use either ***Code A*** or ***Code B***. Further, what we are envisaging is the property by which a sequence of code words is uniquely and instantaneously decodable even if there is no spacing between successive words. The common English words do not posses this property. For example the words " ***FOUND***", " ***AT***" and " ***ION***" when transmitted without spacing yield, at the receiver, an altogether new word" ***FOUNDATION***"! A sufficient condition for such property is that
"No encoded word can be obtained from each other by the addition of more letters " . This property is called " ***prefix property***".

Let $X_k = x_{k1}x_{k2}....x_{km}$, be a code word of some source symbol s_k. Then the sequences of code symbols, $(x_{k1}x_{k2}....x_{kj})$, $\forall\ j \le m$, are called "prefixes" of the code word. Notice tha t a code word of length ' ***m***' will have ' ***m***' prefixes. For example, the code word ***0111*** has four prefixes, viz; ***0***, ***01***, ***011*** and ***0111***.The complete code word is also regarded as a prefix.

Prefix property*: *"A necessary and sufficient condition for a code t o be 'instantaneous' is that no complete code word be a prefix of some other code word".

The *sufficiency* condition follows immediately from the definition of the word " *Instantaneous*". If no word is a prefix of some other word, we can decode any received sequence of code symbols comprising of code words in a direct manner. We scan the received sequence until we come to subsequence which corresponds to a complete code word. Since by assumption it is not a prefix of any other code word, the decoding is unique and there will be no time wasted in the process of decoding. The " *necessary*" condition can be verified by assuming the contrar y and deriving its "contradiction". That is, assume that there exists some word of our code, say x_i, which is a prefix of some other code word x_j. If we scan a received sequence and arrive at a subsequence that corresponds to x_i, this subsequences may be a complete code word or it may just be the first part of code word x_j. We cannot possibly tell which of these alternatives is true until we examine some more code symbols of the sequence. Accordingly, there is definite time wasted before a decision can be made and hence the code is not instantaneous.

5. Optimal codes:

An instantaneous code is said to be optimal if it has " ***minimum average word length***", for a source with a given probability assignment for the source symbols. In such codes, source symbols with higher probabilities of occurrence are made to correspond to shorter code words. Suppose that a source symbol s_i has a probability of occurrence P_i and has a code word of length l_i assigned to it, while a source symbol s_j with probability P_j has a code word of length l_j. If $P_i > P_j$ then let $l_i < l_j$. For the two code words considered, it then follows, that the average length L_1 is given by

$$L_1 = P_i l_i + P_j l_j \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6.3)$$

Now, suppose we interchange the code words so that the code word of length l_j corresponds to s_i and that of length l_i corresponds to s_j. Then, the average length becomes

$$L_2 = P_i l_j + P_j l_i \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6.4)$$

It then follows, $L_2 - L_1 = P_i (l_j - l_i) + P_j (l_i - l_j)$

$$= (P_i - P_j)(l_j - l_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6.5)$$

Since by assumption $P_i > P_j$ and $l_i < l_j$, it is clear that $(L_2 - L_1)$ is positive. That is assignment of source symbols and code word length corresponding to the average length L_1 is shorter, which is the requirement for optimal codes.

A code that satisfies all the five properties is called an " ***irreducible code***".

All the above properties can be arranged as shown in Fig 5.1 which serves as a quick reference of the basic requirements of a code. Fig 5.2 gives the requirements in the form of a 'Tree' diagram. Notice that both sketches illustrate one and the same concept.

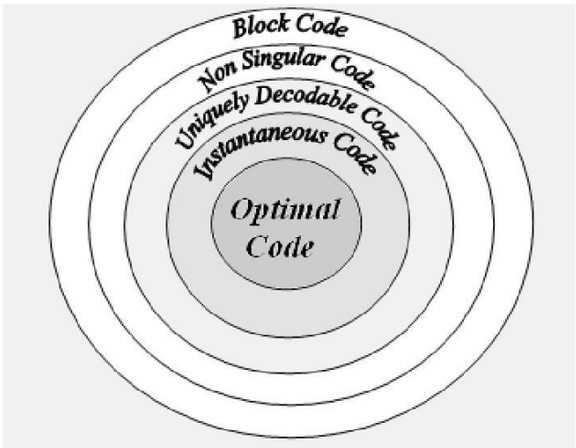


Fig 6.1 Codes - Sub grouping

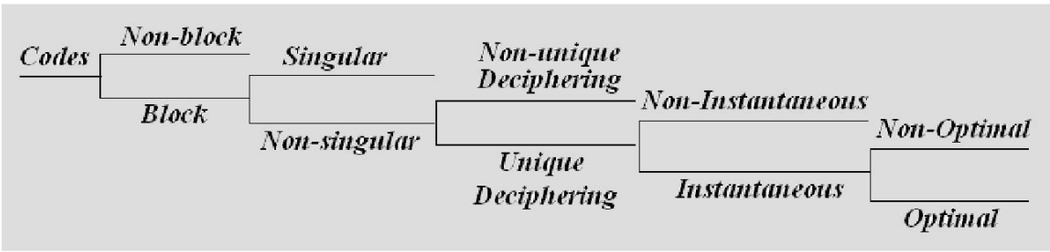


Fig 6.2 Codes - Sub grouping : Indicated as a Tree of Properties

6.3 Construction of Instantaneous Codes:

Consider encoding of a 5 symbol source into Binary instantaneous codes i.e.

$S = \{s_1, s_2, s_3, s_4, s_5\}$; $X = \{0, 1\}$

We may start by assigning '*0*' to s_1

i.e. $s_1 \rightarrow 0$

If this is the case, to have prefix property, all other source symbols must correspond to code words beginning with *1*. If we let s_2 correspond to '*1*', we would be left with no code symbol for encoding the remaining three source symbols. We might have

$s_2 \rightarrow 10$

This in turn would require the remaining code words to start with *11*. If

$s_3 \rightarrow 110$;

then the only 3 bit prefix unused is *111* and we might set

$s_4 \rightarrow 1110$

$s_5 \rightarrow 1111$

In the above code, notice that the starting of the code by letting s_1 correspond ' *0*' has cut down the number of possible code words. Once we have taken this step, we are restricted to code words starting with ' *1*'. Hence, we might expect to have more freedom if we select a ***2-binit*** code word for s_1. We now have four prefixes possible ***00, 01, 10*** and ***11***; the first three can be directly assigned to s_1, s_2 and s_3. With the last one we construct code words of length *3*. Thus the possible instantaneous code is

$s_1 \rightarrow 00$
$s_2 \rightarrow 01$
$s_3 \rightarrow 10$
$s_4 \rightarrow 110$
$s_5 \rightarrow 111$

Thus, observe that shorter we make the first few code words, the longer we will have to make the later code words.

One may wish to construct an instantaneous code by pre-specifying the word lengths. The necessary and sufficient conditions for the existence of such a code are provided by the ***'Kraft Inequality'*** .

6.3.1 Kraft Inequality:

Given a source S = $\{s_1, s_2 \ldots s_q\}$.Let the word lengths of the codes corresponding to these symbols be $l_1, l_2 \ldots\ldots l_q$ and let the code alphabet be X = $\{x_1, x_2 \ldots x_r\}$. Then, an instantaneous code for the source exists iffy

$$\sum_{k=1}^{q} r^{-l_k} \leq 1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (6.6)$$

Eq (6.6) is called ***Kraft Inequality*** (Kraft – 1949).

Example 6.5:

A six symbol source is encoded into Binary codes shown below. Which of these codes are instantaneous?

Source symbol	*Code A*	*Code B*	*Code C*	*Code D*	*Code E*
s_1	*0 0*	*0*	*0*	*0*	*0*
s_2	*0 1*	*1 0 0 0*	*1 0*	*1 0 0 0*	*1 0*
s_3	*1 0*	*1 1 0 0*	*1 1 0*	*1 1 1 0*	*1 1 0*
s_4	*1 1 0*	*1 1 1 0*	*1 1 1 0*	*1 1 1*	*1 1 1 0*
s_5	*1 1 1 0*	*1 1 0 1*	*1 1 1 1 0*	*1 0 1 1*	*1 1 1 1 0*
s_6	*1 1 1 1*	*1 1 1 1*	*1 1 1 1 1*	*1 1 0 0*	*1 1 1 1*
$\sum_{k=1}^{6} 2^{-l_k}$	*1*	$\frac{13}{16} < 1$	*1*	$\frac{7}{8} < 1$	$1\frac{1}{32} > 1$

As a first test we apply the Kraft Inequality and the result is accordingly tabulated. ***Code E does not satisfy Kraft Inequality and it is not an instantaneous code***.

Next we test the prefix property. For ***Code D***, notice that the complete code word for the symbol s_4 is a prefix of the code word for the symbol s_3. Hence it is not an instantaneous code. However, ***Code A***, ***Code B*** and ***Code C*** satisfy the prefix property and are therefore they are instantaneous codes.

Example 6.6:

Given S={$s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9$} and $X=\{0,1\}$. Further if $l_1=l_2=2$ and

$l_3=l_4=l_5=l_6=l_7=l_8=l_9=k$. Then from Kraft inequality, we have

$$2^{-2}+2^{-2}+7\times 2^{-k}\leq$$

$$1\ 7\times 2^{-k}\leq 0.5 \text{ or } \frac{7}{0.5}\leq 2^{-k}$$

$$\therefore \quad k\geq \log_2 14=3.807 \qquad \therefore \quad k_{min}=4$$

Clearly, if $k<4$, it is not possible to construct an instantaneous binary code. Thus if $k\geq 4$, Kraft inequality tells us that an instantaneous code does exist but does not tell us how to construct such a code. The codes for the symbols when $k=4$ are shown below:

s_1 0 0	s_4 1 0 0 1	s_7 1 1 0 0
s_2 0 1	s_5 1 0 1 0	s_8 1 1 1 0
s_3 1 0 0 0	s_6 1 0 1 1	s_9 1 1 1 1

6.3.2 McMillan's Inequality:

Since Instantaneous codes form a sub-class of uniquely decodable codes, we can construct a uniquely decodable code with word lengths $l_1, l_2 \ldots l_q$ satisfying the Kraft inequality. McMillan, (1956) first proved the "necessary part" of the inequality and t he inequality is also called by his name.

Both inequalities are one and the same. Only difference is in the approach to derive the inequality. Since the ideas were totally different and derived independently, the inequality is famous, now-a-days, as " ***Kraft-McMillan inequality***" or the ***K-M inequality***. The two implications of the ***K-M*** inequality are:

(i) Given a set of word lengths, is it possible to construct an instantaneous code?

Yes-if and only if the word lengths satisfy the ***K-M inequality***.

(ii) Whether an already existing code is uniquely decodable?

Yes – if and only if it satisfies the ***K-M inequality***.

Observe the importance of the second implication- For a code to be uniquely decodable its n^{th} extension should be nonsingular. It is not possible to construct all extensions to test this property. Just apply the K-M inequality!

6.3.3 Code Efficiency and Redundancy:

Consider a zero memory source, ***S*** with ***q***-symbols {$s_1, s_2 \ldots s_q$} and symbol probabilities {$p_1, p_2 \ldots p_q$} respectively. Let us encode these symbols into ***r***- ary codes (Using a code alphabet of

r- symbols) with word lengths $l_1, l_2 \ldots l_q$ .We shall find a lower bound for the average length of the code words and hence define efficiency and redundancy of the code.

Let $Q_1, Q_2 \ldots Q_q$ be any set of numbers such that $Q_k \geq 0$ and $\sum_{k=1}^{q} Q_k = 1$. Consider the quantity

$$H(S) - \sum_{k=1}^{q} p_k \log \frac{1}{Q_k}$$

$$= \sum_{k=1}^{q} p_k \log \frac{1}{p_k} - \sum_{k=1}^{q} p_k \log \frac{1}{Q_k} = \sum_{k=1}^{q} p_k \log \frac{Q_k}{p_k}$$

$$= \log e \sum_{k=1}^{q} p_k \ln \frac{Q_k}{p_k} \leq \log e \sum_{k=1}^{q} p_k \left(\frac{Q_k}{p_k} - 1 \right) = \log e \sum_{k=1}^{q} Q_k - \sum_{k=1}^{q} p_k \leq 0$$

$$\ldots\ldots\ldots (\text{as } \sum_{k=1}^{q} P_k = 1 \text{ and } \sum_{k=1}^{q} Q_k = 1, \text{by assumption})$$

$$\text{Thus it follows that } \quad H(S) \leq \sum_{k=1}^{q} P_k \log \frac{1}{Q_k} \qquad \ldots\ldots\ldots\ldots (6.7)$$

Equality holds iffy $Q_k = p_k$. Eq. (5.21) is valid for any set of numbers Q_k that are non negative and sum to unity. We may then choose:

$$Q_k = \frac{r^{-l_k}}{\sum_{k=1}^{q} r^{-l_k}} \qquad \ldots\ldots\ldots\ldots (6.8)$$

and obtain

$$H(s) \leq \sum_{k=1}^{q} p_k \log r^{l_k} \sum_{k=1}^{q} r^{-l_k}$$

$$\leq \sum_{k=1}^{q} p_k l_k \log r + \log \sum_{k=1}^{q} r^{-l_k}$$

$$i.e \quad H(S) \leq \log r \sum_{k=1}^{q} p_k l_k + \log \sum_{k=1}^{q} r^{-l_k} \qquad \ldots\ldots\ldots\ldots (6.9)$$

Defining

$$L = \sum_{k=1}^{q} p_k l_k \qquad \ldots\ldots\ldots\ldots (6.10)$$

Which gives the average length of the code words, and observing that the second term in Eq.(5.23) is either negative or at most zero (as from Kraft Inequality $\sum_{k=1}^{q} r^{-l_k} \leq 1$ and logarithm of a number which is less than unity is always negative) we conclude:

$$H(S) \leq L \log r \qquad \ldots\ldots\ldots\ldots (6.11)$$

$$\text{Or } L \geq \frac{H(S)}{\log r} \qquad \ldots\ldots\ldots\ldots (6.12)$$

Notice that achievement of the lower bound is possible for the particular case when:

i) $\sum_{k=1}^{q} r^{-l_k} = 1$ and (ii) $p_k = r^{-l_k}$

As, if this is the case Q_k
that:

$l_k = -\log_r \frac{1}{p_k} = p_k$, hence for the equality to hold it follows that we must choose l_k such

…… (6.13) and

l_k must be an integer for all $k = 1, 2 \ldots q$.

Eq. (6.11) can be re-written as:

$$\frac{H(S)}{L} \leq \log r \qquad \ldots\ldots\ldots\ldots\ldots \qquad (6.14)$$

The LHS of Eq. (6.14) is simply (The entropy of the source in bits per source symbol) ÷ (no. of code symbols per source symbol) or bits per code symbol; which is nothing but the actual entropy of the code symbols. RHS is the maximum value of this entropy when the code symbols are all equi-probable. Thus we can define the code efficiency based on Eq. (6.14) as below:

" ***Code efficiency is the ratio of the average information per symbol of the encoded language to the maximum possible information per code symbol***". Mathematically, we write

Code efficiency, $\eta_c \triangleq \frac{H(S)}{L} : \log r$

Or $$\eta_c \triangleq \frac{H(S)}{L \log r} \qquad \ldots\ldots\ldots\ldots \qquad (6.15)$$

Accordingly, Redundancy of the code, $E_c = 1 - \eta_c$ …… (6.16).

Example 6.7:

Let the source have four messages $S = \{s_1, s_2, s_3, s_4\}$ with $P = \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right\}$

$$\therefore H(S) = \frac{1}{2}\log 2 + \log 4 + 2 \times \frac{1}{8}\log 8 = 1.75 \text{ bits/sym.}$$

If S itself is assumed to be the alphabet then we have

$L = 1, r = 4$ and $\eta_c = \frac{1.75}{\log 4} = 0.875$, i.e. $\eta_c = 87.5\%$, and $E_c = 12.5\%$

Suppose the messages are encoded into a binary alphabet, $X = \{0, 1\}$ as

	p_k	Code	l_k
s_1	1/2	0	$l_1 = 1$
s_2	1/4	1 0	$l_2 = 2$
s_3	1/8	1 1 0	$l_3 = 3$
s_4	1/8	1 1 1	$l_4 = 3$

$$\text{We have } L = \sum_{k=1}^{4} l_k p_k = 1.\frac{1}{2} + 2.\frac{1}{4} + 3.\frac{1}{8} + 3.\frac{1}{8} = 1.75 \text{binits/symbol}$$

Since $r=2$, $\eta_c = \frac{H(S)}{L \log r} = \frac{1.75}{1.75 \log_2 2} = 1$

i.e. $\eta_c = 100\%$, and hence $Ec = 1 - \eta_c = 0\%$

Thus by proper encoding, the efficiency can be increased.

Example 6.8:

Suppose the probabilities of the messages in Example 6.7 are changed to $P = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}\right)$

Then, $H(S) = 2 \times \frac{1}{3} \log 3 + 2 \times \frac{1}{6} \log 6$

$\log 3 + \frac{1}{3} = 1.918$ bits/sym

For the codes listed in Example 6.7, we have:

$L = 1.\frac{1}{3} + 2.\frac{1}{3} + 3.\frac{1}{6} + 3.\frac{1}{6} = 2 \text{ binits/symbol,}$ and $\eta_c = \frac{H(S)}{L \log r} = \frac{1.918}{2 \log_2 2} = 0.959 \text{ or } 95.9\%$

$E_c = 1 - \eta_c = 0.041 \text{ or } 4.1\%$

Notice that in Example 6.7, the equality $L = H(S)/\log r$ is strict since $l_k = \log 1/p_k$ and in Example 6.8, the inequality $L > H(S)/\log r$ is strict as $l_k \neq \log 1/p_k$.

6.3.4 Shannon's First Theorem (Noiseless Coding Theorem):

Eq. (6.12) suggests the lower bound on L, the average word length of the code, expressed as a fraction of code symbols per source symbol. However, we know that each individual code word will have an integer number of code symbols. Thus, we are faced with the problem of what to choose for the value of l_k, the number of code symbols in the code word corresponding to a source symbol s_k,

when the quantity in Eq. (6.13) viz. $l_k = \log_r \frac{1}{p_k}$

the next integer value greater than $\log_r \frac{1}{p_k}$

$$\log_r \frac{1}{p_k} \leq l_k \leq \log_r \frac{1}{p_k} + 1 \quad \ldots\ldots\ldots\ldots\ldots \quad (6.17)$$

Eq. (6.17) also satisfies the Kraft inequality, since the left inequality gives:

$\frac{1}{p_k} \leq r^{l_k}$, or $p_k \geq r^{-l_k}$. from which $\sum_{k=1}^{q} p_k = 1 \geq \sum_{k=1}^{q} r^{-l_k}$

Further , since $\log_r \frac{1}{p_k} = \frac{\log_2 \frac{1}{p_k}}{\log_2 r}$, Eq (6.17) can be re-written as:

$$\frac{\log(1/p_k)}{\log r} \leq l_k \leq \frac{\log(1/p_k)}{\log r} + 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6.18)$$

Multiplying Eq. (6.18) throughout by p_k and summing for all values of k, we have

$$\sum_{k=1}^{q} \frac{p_k \log \frac{1}{p_k}}{\log r} \le \sum_{k=1}^{q} p_k l_k \le \sum_{k=1}^{q} \frac{p_k \log \frac{1}{p_k}}{\log r} + \sum_{k=1}^{q} p_k \text{ , or}$$

$$\frac{H(S)}{\log r} \le L \le \frac{H(S)}{\log r} + 1 \quad \text{.....} \quad (6.19)$$

To obtain better efficiency, one will use the n^{th} extension of S, giving L_n as the new average word length. Since Eq. (5.33) is valid for any zero- memory source, it is also valid for S^n, and hence, we have

$$\frac{H(S^n)}{\log r} \le L_n < \frac{H(S^n)}{\log r} + 1 \quad \text{.....} \quad (6.20)$$

Since $H(S^n) = n\,H(S)$, Eq. (6.20 reduces to

$$\frac{H(S)}{\log r} \le \frac{L_n}{n} < \frac{H(S)}{\log r} + \frac{1}{n} \quad \text{.....} \quad (6.21)$$

It follows from Eq. (6.21) with n $\to \infty$

$$\lim_{n \to \infty} \frac{L_n}{n} = \frac{H(S)}{\log r} \quad \text{.....} \quad (6.22)$$

Here L_n/n is the average number of code alphabet symbols used per single symbol of S, when the input to the encoder is n-symbol message for the extended source S^n. But $\frac{L_n}{n} \neq L$, where L is the average word length for the source S and in general $\frac{L_n}{n} \le L$. The code capacity is now $\frac{L_n}{n} \log r = C$ bits/message of the channel and for successful transmission of messages through the channel we have:

$$H(S) \le \frac{L_n}{n} \log r = C \text{ bits/message} \quad \text{.....} \quad (6.23)$$

Eq. (6.21) is usually called the " ***Noiseless coding Theorem***" and is the essence of " ***Shannon's First Fundamental Theorem***". Notice that in the above discussions we have not considered any effects of noise on the codes. The emphasis is only on how to most efficiently encode our source. The theorem may be stated as belo

CYCLIC CODES

"***Binary cyclic codes"*** form a sub class of linear block codes. Majority of important linear block codes that are known to-date are either cyclic codes or closely related to cyclic codes. Cyclic codes are attractive for two reasons: First, encoding and syndrome calculations can be easily

implemented using simple shift registers with feed back connections. Second, they posses well defined mathematical structure that permits the design of higher-order error correcting codes.

A binary code is said to be "***cyclic***" if it satisfies:

1. *Linearity property* – sum of two code words is also a code word.
2. *Cyclic property* – Any lateral shift of a code word is also a code word.

The second property can be easily understood from Fig, 7.1. Instead of writing the code as a row vector, we have represented it along a circle. The direction of traverse may be either clockwise or counter clockwise (right shift or left shift).

For example, if we move in a counter clockwise direction then starting at ' ***A***' the code word is ***110001100*** while if we start at ***B*** it would be ***011001100***. Clearly, the two code words are related in that one is obtained from the other by a cyclic shift.

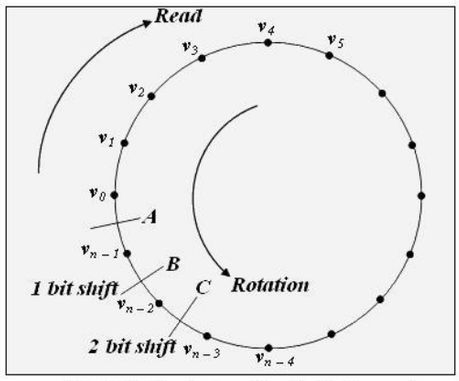


Fig 7.1 Illustrating the Cyclic Property.

If the ***n*** - tuple, read from ' ***A***' in the ***CW*** direction in Fig 7.1,

$$v = (v_0, v_1, v_2, v_3, v_{n-2}, v_{n-1}) \quad \text{..... (7.1)}$$

is a code vector, then the code vector, read from ***B***, in the ***CW*** direction, obtained by a one bit cyclic right shift:

$$v^{(1)} = (v_{n-1}, v_0, v_1, v_2, \ldots v_{n-3}, v_{n-2},) \quad \text{.....(7.2)}$$

is also a code vector. In this way, the ***n*** - tuples obtained by successive cyclic right shifts:

$$v^{(2)} = (v_{n-2}, v_{n-1}, v_n, v_0, v_1 \ldots v_{n-3}) \quad \text{..... (7.3 } a)$$

$$v^{(3)} = (v_{n-3}, v_{n-2}, v_{n-1}, v_n, \ldots v_0, v_1, v_{n-4}) \quad \text{.....(7.3 } b)$$

M

$$v^{(i)} = (v_{n-i}, v_{n-i+1}, \ldots v_{n-1}, v_0, v_1, \ldots v_{n-i-1}) \quad \text{..... (7.3 } c)$$

are all code vectors. This property of cyclic codes enables us to treat the elements of each code vector as the co-efficients of a polynomial of degree ***(n-1).***

This is the property that is extremely useful in the analysis and implementation of these codes. Thus we write the "code polynomial' ***V(X)*** for the code in Eq (7.1) as a vector polynomial as:

$$V(X) = v_0 + v_1 X + v_2 X^2 + v_3 X^3 + \ldots + v_{i-1} X^{i-1} + \ldots + v_{n-3} X^{n-3} + v_{n-2} X^{n-2} + v_{n-1} X^{n-1} \quad \ldots\ldots\ldots (7.4)$$

Notice that the co-efficients of the polynomial are either '*0*' or '*1*' (binary codes), i.e. they belong to ***GF (2)*** as discussed in sec 6.7.1.

. Each power of ***X*** in ***V(X)*** represents a one bit cyclic shift in time.

. Therefore multiplication of ***V(X)*** by ***X*** maybe viewed as a cyclic shift or rotation to the right subject to the condition $X^n = 1$. This condition (i) restores ***XV(X)*** to the degree ***(n-1)*** (ii) Implies that right most bit is fed-back at the left.

. This special form of multiplication is called "***Multiplication modulo*** " $X^n + 1$"

. Thus for a single shift, we have

$$XV(X) = v_0 X + v_1 X^2 + v_2 X^3 + \ldots\ldots\ldots + v_{n-2} X^{n-1} + v_{n-1} X^n$$

$$(+ v_{n-1} + v_{n-1}) \ldots \text{(Manipulate } A + A = 0 \text{ Binary Arithmetic)}$$

$$= v_{n-1} + v_0 X + v_1 X^2 + + v_{n-2} X^{n-1} + v_{n-1}(X^n + 1)$$

$= V^{(1)}(X)$ = Remainder obtained by dividing ***XV(X)*** by $X^n + 1$
(Remember: ***X*** mod ***Y*** means remainder obtained after dividing ***X*** by ***Y***)

Thus it turns out that

$$V^{(1)}(X) = v_{n-1} + v_0 X + v_1 X^2 + \ldots\ldots + v_{n-2} X^{n-1} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.5)$$

I

is the code polynomial for $v^{(1)}$. We can continue in this way to arrive at a general format:

$$X^i V(X) = V^{(i)}(X) + q(X)(X^n + 1) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.6)$$

Remainder ***Quotient***

Where

$$V^{(i)}(X) = v_{n-i} + v_{n-i+1} X + v_{n-i+2} X^2 + \ldots v_{n-1} X^i + \ldots v_0 X^{i-1} + v_1 X^{i+1} + \ldots v_{n-i-2} X^{n-2} + v_{n-i-1} X^{n-} \ldots\ldots\ldots (7.7)$$

7.1 Generator Polynomial for Cyclic Codes:

An ***(n, k)*** cyclic code is specified by the complete set of code polynomials of degree ≤ ***(n-1)*** and contains a polynomial ***g(X)***, of degree ***(n-k)*** as a factor, called the "***generator polynomial***" of the code. This polynomial is equivalent to the generator matrix ***G***, of block codes. Further, it is the only polynomial of minimum degree and is unique. Thus we have an important theorem

Theorem 7.1 "If ***g(X)*** is a polynomial of degree ***(n-k)*** and is a factor of $(X^n + 1)$ then ***g(X)*** generates an ***(n, k)*** cyclic code in which the code polynomial ***V(X)*** for a data vector $u = (u_0, u_1 \ldots u_{k-1})$ is generated by

$$V(X) = U(X) \times g(X) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.8)$$

Where $U(X) = u_0 + u_1 X + u_2 X^2 + ... + u_{k-1} X^{k-1}$ ..... (7.9)

is the data polynomial of degree ***(k-1)***.

The theorem can be justified by Contradiction: - If there is another polynomial of same degree, then add the two polynomials to get a polynomial of degree < ***(n, k)*** (use linearity property and binary arithmetic). Not possible because minimum degree is (***n-k***). Hence ***g(X)*** is unique

Clearly, there are 2^k code polynomials corresponding to 2^k data vectors. The code vectors corresponding to these code polynomials form a linear (***n, k***) code. We have then, from the theorem

$$g(X) = 1 + \sum_{i=1}^{n-k-1} g_i X^i + X^{n-k} \quad \text{.....} \quad (7.10)$$

$$\text{As } g(X) = g_0 + g_1 X + g_2 X^2 + \text{.....} + g_{n-k-1} X^{n-k-1} + g_{n-k} X^{n-k} \quad \text{.....} \quad (7.11)$$

is a polynomial of minimum degree, it follows that $g_0 = g_{n-k} = 1$ always and the remaining co-efficients may be either' ***0***' of '***1***'. Performing the multiplication said in Eq (7.8) we have:

$$U(X)\, g(X) = u_0\, g(X) + u_1 X\, g(X) + \ldots + u_{k-1} X^{k-1} g(X) \quad \text{.....} \quad (7.12)$$

Suppose $u_0 = 1$ and $u_1 = u_2 = \ldots = u_{k-1} = 0$. Then from Eq (7.8) it follows ***g(X)*** is a code word polynomial of degree ***(n-k)***. This is treated as a ' ***basis code polynomial***' (All rows of the ***G*** matrix of a block code, being linearly independent, are also valid code vectors and form ' ***Basis vectors***' of the code). Therefore from cyclic property $X^i\, g(X)$ is also a code polynomial. Moreover, from the linearity property - a linear combination of code polynomials is also a code polynomial. It follows therefore that any multiple of ***g(X)*** as shown in Eq (7.12) is a code polynomial. Conversely, any binary polynomial of degree ≤ ***(n-1)*** is a code polynomial if and only if it is a multiple of ***g(X).*** The code words generated using Eq (7.8) are in non-systematic form. Non systematic cyclic codes can be generated by simple binary multiplication circuits using shift registers. .

In this book we have described cyclic codes with right shift operation. Left shift version can be obtained by simply re-writing the polynomials. Thus, for left shift operations, the various polynomials take the following form

$$U(X) = u_0 X^{k-1} + u_1 X^{k-2} + \ldots\ldots + u_{k-2} X + u_{k-1} \quad \text{.....} \quad (7.13\ a)$$

$$V(X) = v_0 X^{n-1} + v_1 X^{n-2} + \ldots + v_{n-2} X + v_{n-1} \quad \text{.....} \quad (7.13\ b)$$

$$g(X) = g_0 X^{n-k} + g_1 X^{n-k-1} + \ldots + g_{n-k-1} X + g_{n-k} \quad \text{.....} \quad (7.13\ c)$$

$$= X^{n-k} + \sum_{i=1}^{n-k} g_i X^{n-k-i} + g_{n-k} \quad \text{.....} \quad (7.13d)$$

Other manipulation and implementation procedures remain unaltered.

7.2 Multiplication Circuits:

Construction of encoders and decoders for linear block codes are usually constructed with combinational logic circuits with mod-2 adders. Multiplication of two polynomials ***A(X)*** and ***B(X)*** and the division of one by the other are realized by using sequential logic circuits, mod-2 adders and shift registers. In this section we shall consider multiplication circuits.

As a convention, the higher-order co-efficients of a polynomial are transmitted first. This is the reason for the format of polynomials used in this book.

For the polynomial: $A(X) = a_0 + a_1 X + a_2 X^2 + ... + a_{n-1}X^{n-1}$ …… (7.14)

where a_i**'s** are either a '***0***' or a '***1***', the right most bit in the sequence $(a_0, a_1, a_2 \ldots a_{n-1})$ is transmitted first in any operation. The product of the two polynomials ***A(X)*** and ***B(X)*** yield:

$$C(X) = A(X) \times B(X)$$

$$= (a_0 + a_1 X + a_2 X^2 + \ldots \ldots\ldots\ldots\ldots + a_{n-1}X^{n-1})\,(b_0 + b_1 X + b_2 X^2 + \ldots + b_{m-1} X^{m-1})$$

$$= a_0b_0 + (a_1b_0 + a_0b_1) X + (a_0b_2 + b_0a_2 + a_1b_1) X^2 + \ldots + (a_{n-2}b_{m-1} + a_{n-1}b_{m-2}) X^{n+m-3} + a_{n-1}b_{m-1}X^{n+m-2}$$

This product may be realized with the circuits of Fig 7.2 (*a*) or (*b*), where ***A(X)*** is the input and the co-efficient of ***B(X)*** are given as weighting factor connections to the mod - 2 .adders. A '***0***' indicates no connection while a '***1***' indicates a connection. Since higher order co-efficients are first sent, the highest order co-efficient $a_{n-1} \times b_{m-1}$ of the product polynomial is obtained first at the output of Fig 7.2(*a*). Then the co-efficient of X^{n+m-3} is obtained as the sum of $\{a_{n-2}b_{m-1} + a_{n-1}\, b_{m-2}\}$, the first term directly and the second term through the shift register ***SR1***. Lower order co-efficients are then generated through the successive ***SR***'s and mod-2 adders. After (***n + m - 2***) shifts, the ***SR***'s contain {***0, 0… 0, a*** $_0$***, a*** $_1$} and the output is ($a_0\, b_1 + a_1\, b_0$) which is the co-efficient of ***X.*** After ***(n + m-1)*** shifts, the SR's contain (***0, 0, 0,0,*** a_0) and the out put is $a_0 \times b_0$. The product is now complete and the contents of the ***SR***'s become (***0, 0, 0 …0, 0***). Fig 7.2(*b*) performs the multiplication in a similar way but the arrangement of the ***SR***'s and ordering of the co-efficients are different (reverse order!). This modification helps to combine two multiplication operations into one as shown in Fig 7.2(*c*).

From the above description, it is clear that a non-systematic cyclic code may be generated using (***n-k***) shift registers. Following examples illustrate the concepts described so far.

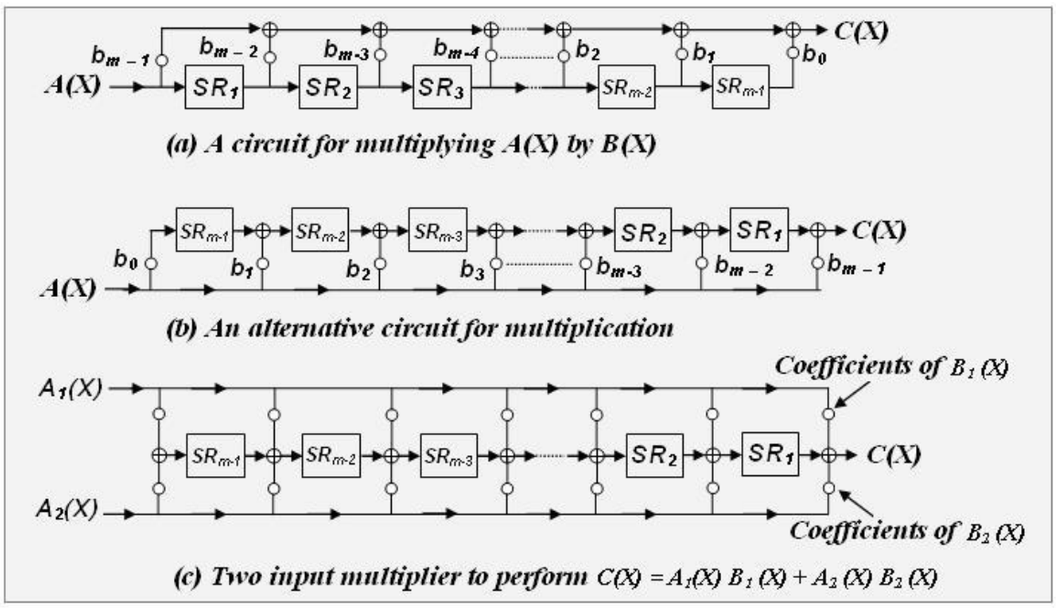


Fig 7.2 Multiplication circuits

Example 7.1: Consider that a polynomial ***A(X)*** is to be multiplied by

$$B(X) = 1 + X + X^3 + X^4 + X^6$$

The circuits of Fig 7.3 (*a*) and (*b*) give the product ***C(X) = A(X). B(X)***

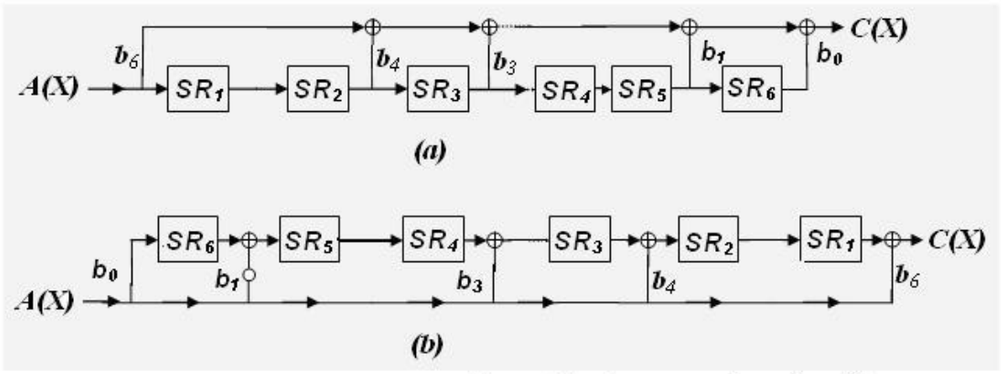


Fig 7.3 Circuits to perform C(X) = A(X) × (1 + X + X³ + X⁴ + X⁶)

Example 7.2: Consider the generation of a (***7, 4***) cyclic code. Here ***(n- k)*** = (***7-4***) = ***3*** and we have to find a generator polynomial of degree ***3*** which is a factor of $X^n + 1 = X^7 + 1$.

To find the factors of' degree ***3***, divide X^7+1 by X^3+aX^2+bX+1, where '***a***' and '***b***' are binary numbers, to get the remainder as $abX^2 + (1 + a + b)X + (a+b+ab+1)$. Only condition for the remainder to be zero is ***a +b=1*** which means either ***a = 1, b = 0*** or ***a = 0, b = 1***. Thus we have two possible polynomials of degree 3, namely

$$g_1(X) = X^3 + X^2 + 1 \text{ and } g_2(X) = X^3 + X + 1$$

In fact, $X^7 + 1$ can be factored as:

$$(X^7+1) = (X+1)\,(X^3+X^2+1)\,(X^3+X+1)$$

Thus selection of a '*good*' generator polynomial seems to be a major problem in the design of cyclic codes. No clear-cut procedures are available. Usually computer search procedures are followed.

Let us choose $g(X) = X^3 + X + 1$ as the generator polynomial. The encoding circuits are shown in Fig 7.4(*a*) and (*b*).

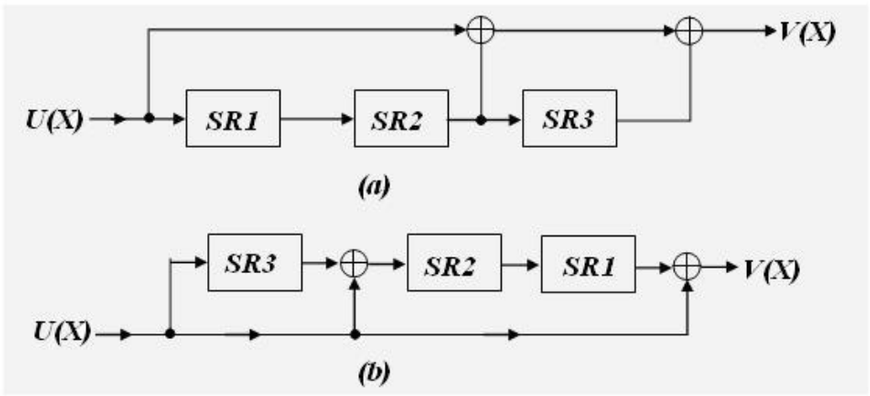


Fig 7.4 Generation of Non-systematic Cyclic codes V(X) = U(X).g(X)

To understand the operation, Let us consider $u = (10\ 1\ 1)$ i.e.

$$U(X) = 1 + X^2 + X^3.$$

We have $V(X) = (1 + X^2 + X^3)\,(1 + X + X^3)$.

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

$$= 1 + X + X^2 + X^3 + X^4 + X^5 + X^6 \quad \text{because } (X^3 + X^3 = 0)$$

$$\Rightarrow v = (1\ 1\ 1\ 1\ 1\ 1\ 1)$$

The multiplication operation, performed by the circuit of Fig 7.4(*a*), is listed in the Table below step by step. In shift number 4, ' ***000'*** is introduced to flush the registers. As seen from the tabulation the product polynomial is:

$$V(X) = 1 + X + X^2 + X^3 + X^4 + X^5 + X^6,$$

and hence out put code vector is $v = (1\ 1\ 1\ 1\ 1\ 1\ 1)$, as obtained by direct multiplication. The reader can verify the operation of the circuit in Fig 7.4(*b*) in the same manner. Thus the multiplication circuits of Fig 7.4 can be used for generation of non-systematic cyclic codes.

Table showing sequence of computation

Shift Number	Input Queue	Bit shifted IN	Contents of shift registers. SRI	SR2	SR3	Out put	Remarks
0	0001011	-	0	0	0	-	Circuit In reset mode
1	000101	1	1	0	0	1	Co-efficient of X^0
2	00010	1	1	1	0	1	Co-efficient of X^3
3	0001	0	0	1	1	1	X^4 co-efficient
*4	000	1	1	0	1	1	X^5 co-efficient
5	00	0	0	1	0	1	X^2 co-efficient
6	0	0	0	0	1	1	X^1 co-efficient
7	-	0	0	0	0	1	X^0 co-efficient

7.3 Dividing Circuits:

As in the case of multipliers, the division of ***A (X)*** by ***B (X)*** can be accomplished by using shift registers and Mod-2 adders, as shown in Fig 7.5. In a division circuit, the first co-efficient of the quotient is $(a_{n-1} \div (b_{m-1}) = q_1$, and $q_1.B(X)$ is subtracted from ***A (X)***. This subtraction is carried out by the feed back connections shown. This process will continue for the second and subsequent terms. However, remember that these coefficients are binary coefficients. After ***(n-1)*** shifts, the entire quotient will appear at the output and the remainder is stored in the shift registers.

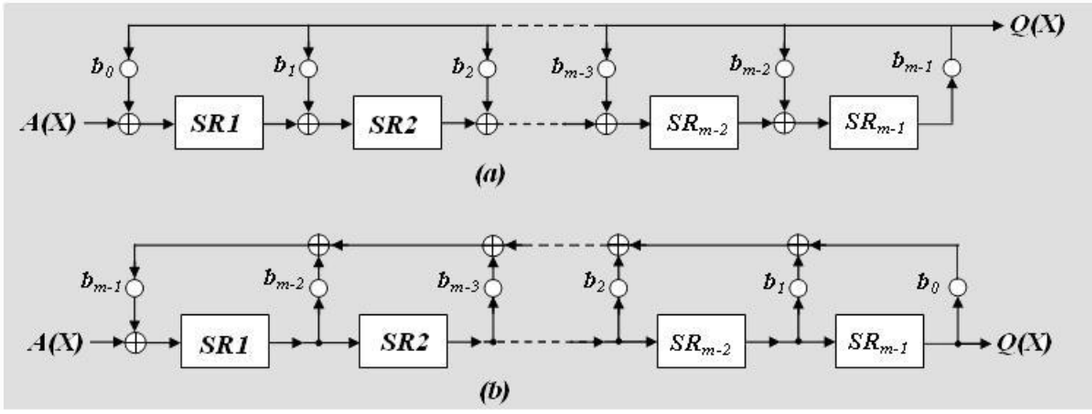


Fig 7.5 Dividing Circuits

It is possible to combine a divider circuit with a multiplier circuit to build a "composite multiplier-divider circuit" which is useful in vari ous encoding circuits. An arrangement to accomplish this is shown in Fig 7.6(*a*) and an illustration is shown in Fig 7.6(*b*).

We shall understand the operation of one divider circuit through an example. Operation of other circuits can be understood in a similar manner.

Example7.3:

Let $A(X) = X^3+X^5+X^6$, $\rightarrow A= (0001011)$, $B(X) = 1 +X+X^3$. We want to find the quotient and remainder after dividing ***A(X)*** by ***B(X).*** The circuit to perform this division is shown in Fig 7.7, drawn using the format of Fig 7.5(*a*). The operation of the divider circuit is listed in the table:

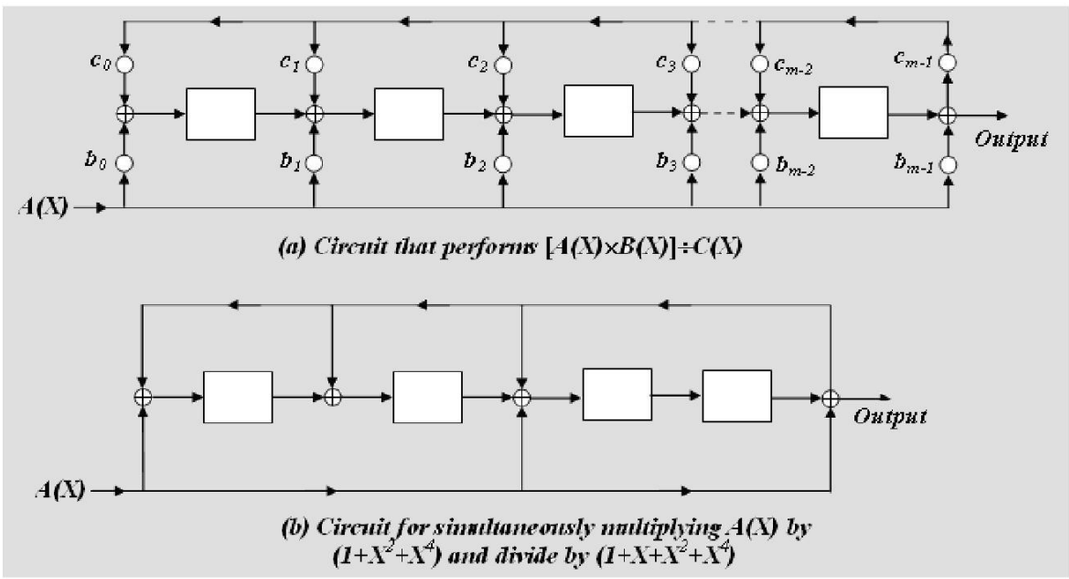


Fig 7.6 Circuits for Simultaneous Multiplication and Division

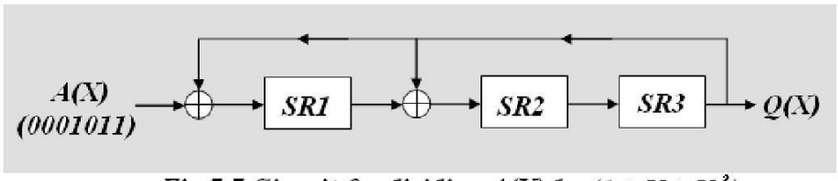


Fig 7.7 Circuit for dividing A(X) by $(1+X+X^3)$

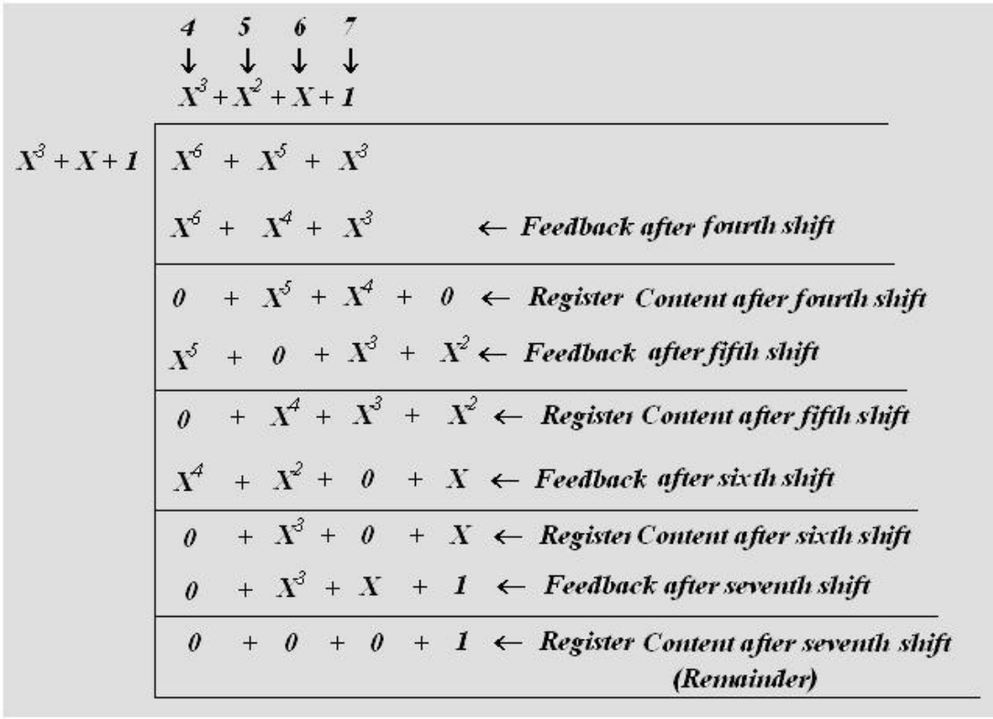
Table Showing the Sequence of Operations of the Dividing circuit

Shift Number	Input Queue	Bit shifted IN	Contents of shift Registers.			Out put	Remarks
			SRI	SR2	SR3		
0	0001011	-	0	0	0	-	Circuit in reset mode
1	000101	1	1	0	0	0	Co-efficient of X^0
2	00010	1	1	1	0	0	Co-efficient of X^5
3	0001	0	0	1	1	0	X^4 co-efficient
4	*000	1	0	1	1	1	X^3 co-efficient
5	00	0	1	1	1	1	X^2 co-efficient
6	0	0	1	0	1	1	X^1 co-efficient
7	-	0	1	0	0	1	X^0 co-efficient

The quotient co-efficients will be available only after the fourth shift as the first three shifts result in entering the first 3-bits to the shift registers and in each shift out put of the last register, ***SR3***, is zero.

The quotient co-efficient serially presented at the out put are seen to be ***(1111)*** and hence the quotient polynomial is $Q(X) = 1 + X + X^2 + X^3$. The remainder co-efficients are ***(1 0 0)*** and the remainder polynomial is ***R(X) = 1***. The polynomial division steps are listed in the next page.

Division Table for Example 7.3:



NOTE: Read the register contents at each step in the direction of the arrow shown, (three co-efficients only)

7.4 Systematic Cyclic Codes:

Let us assume a systematic format for the cyclic code as below:

$$v = (p_0, p_1, p_2 \ \ldots \ p_{n-k-1}, u_0, u_1, u_2 \ldots u_{k-1}) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.15)$$

The code polynomial in the assumed systematic format becomes:

$$V(X) = p_0 + p_1X + p_2X^2 + \ldots + p_{n-k-1}X^{n-k-1} + u_0X^{n-k} + u_1X^{n-k+1} + \ldots + u_{k-1}X^{n-1} \quad \ldots\ldots\ldots\ldots \quad (7.16)$$

$$= P(X) + X^{n-k}U(X) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.17)$$

Since the code polynomial is a multiple of the generator polynomial we can write:

$$V(X) = P(X) + X^{n-k}U(X) = Q(X)\,g(X) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.18)$$

$$\Rightarrow \frac{X^{n-k}U(X)}{g(X)} = Q(X) + \frac{P(X)}{g(X)} \quad \ldots\ldots\ldots \quad (7.19)$$

Thus division of $X^{n-k}U(X)$ by $g(X)$ gives us the quotient polynomial $Q(X)$ and the remainder polynomial $P(X)$. Therefore to obtain the cyclic codes in the systematic form, we determine the remainder polynomial $P(X)$ after dividing $X^{n-k}U(X)$ by $g(X)$. This division process can be easily achieved by noting that "multiplication by X^{n-k} amounts to shifting the sequence by $(n-k)$ bits". Specifically in the circuit of Fig 7.5(*a*), if the input $A(X)$ is applied to the Mod-2 adder

after the $(n-k)^{th}$ shift register the result is the division of $X^{n-k} A(X)$ by $B(X)$.

Accordingly, we have the following scheme to generate systematic cyclic codes. The generator polynomial is written as:

$$g(X) = 1 + g_1X + g_2X^2 + g_3X^3 + \ldots + g_{n-k-1}X^{n-k-1} + X^{n-k} \quad \ldots\ldots\ldots\ldots \quad (7.20)$$

The circuit of Fig 7.8 does the job of dividing $X^{n-k}U(X)$ by $g(X)$. The following steps describe the encoding operation.

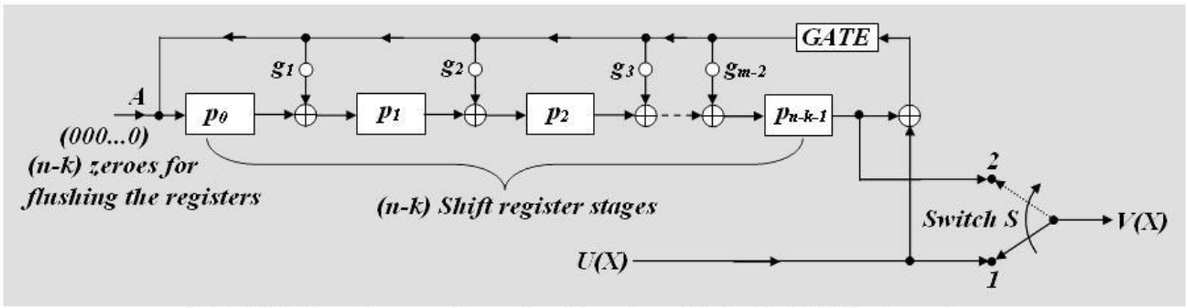


Fig 7.8 Systematic encoding of cyclic codes with (n-k) shift Register stages

1. The switch **S** is in position **1** to allow transmission of the message bits directly to an out put shift register during the first **k**-shifts.
2. At the same time the '**GATE**' is '**ON**' to allow transmission of the message bits into the **(n-k)** stage encoding shift register
3. After transmission of the k^{th} message bit the **GATE** is turned **OFF** and the switch **S** is moved to position **2**.
4. **(n-k)** zeroes introduced at "**A**" after step **3**, clear the encoding register by moving the parity bits to the output register
5. The total number of shifts is equal to **n** and the contents of the output register is the code word polynomial $V(X) = P(X) + X^{n-k}U(X)$.
6. After step-**4**, the encoder is ready to take up encoding of the next message input

Clearly, the encoder is very much simpler than the encoder of an **(n, k)** linear block code and the memory requirements are reduced. The following example illustrates the procedure.

Example 7.4:

Let $u = (1\ 0\ 1\ 1)$ and we want a $(7, 4)$ cyclic code in the systematic form. The generator polynomial chosen is $g(X) = 1 + X + X3$

For the given message, $U(X) = 1 + X^2 + X^3$

$$X^{n-k}U(X) = X^3U(X) = X^3 + X^5 + X^6$$

We perform direct division $X^{n-k}U(X)$ by $g(X)$ as shown below. From direct division observe that $p_0=1, p_1=p_2=0$. Hence the code word in systematic format is:

$$v = (p_0, p_1, p_2; u_0, u_1, u_2, u_3) = (1, 0, 0, 1, 0, 1, 1)$$

$$
\begin{array}{r|l}
 & X^3+X^2+X+1 \\
\hline
X^3+X^2+X+1 & X^6+X^5+X^3 \\
 & X^6+X^4+X^3 \\
\hline
 & X^5+X^4 \\
 & X^5+X^3+X^2 \\
 & X^4+X^3+X^2 \\
 & X^4+X^2+X \\
\hline
 & X^3+X \\
 & X^3+X+1 \\
\hline
 & 1 \quad \text{(Remainder)}
\end{array}
$$

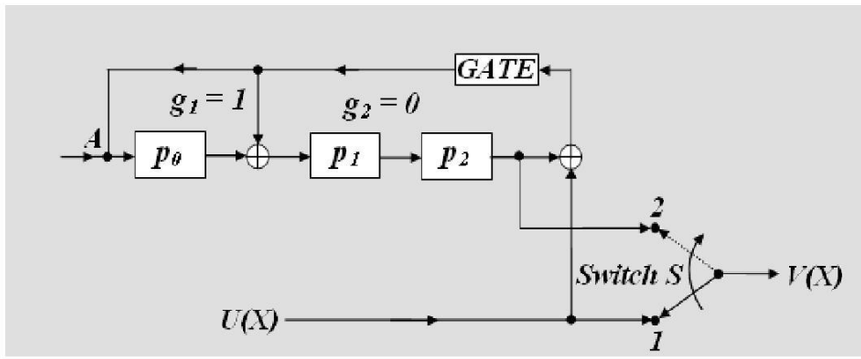


Fig 7.9 Encoder for the (7,4) cyclic code of Example 7.4

The encoder circuit for the problem on hand is shown in Fig 7.9. The operational steps are as follows:

Shift Number	*Input Queue*	*Bit shifted IN*	*Register contents*	*Output*
0	*1011*	*-*	*000*	*-*
1	*101*	*1*	*110*	*1*
2	*10*	*1*	*101*	*1*
3	*1*	*0*	*100*	*0*
4	*-*	*1*	*100*	*1*

After the Fourth shift ***GATE*** Turned ***OFF***, switch ***S*** moved to position ***2,*** and the parity bits contained in the register are shifted to the output. The out put code vector is ***v = (100 1011)*** which agrees with the direct hand calculation.

7.6 Syndrome Calculation - Error Detection and Error Correction:

Suppose the code vector $v = (v_0, v_1, v_2 \ldots v_{n-1})$ is transmitted over a noisy channel. Hence the received vector may be a corrupted version of the transmitted code vector. Let the received code vector be $r = (r_0, r_1, r_2 \ldots r_{n-1})$. The received vector may not be anyone of the 2^k valid code vectors. The function of the decoder is to determine the transmitted code vector based on the received vector.

The decoder, as in the case of linear block codes, first computes the syndrome to check whether or not the received code vector is a valid code vector. In the case of cyclic codes, if the syndrome is zero, then the received code word polynomial must be divisible by the generator polynomial. If the syndrome is non-zero, the received word contains transmission errors and needs error correction. Let the received code vector be represented by the polynomial

$$R(X) = r_0 + r_1X + r_2X^2 + \ldots + r_{n-1}X^{n-1}$$

Let ***A(X)*** be the quotient and ***S(X)*** be the remainder polynomials resulting from the division of ***R(X)*** by ***g(X)*** i.e.

$$\frac{R(X)}{g(X)} = A(X) + \frac{S(X)}{g(X)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.21)$$

The remainder ***S(X)*** is a polynomial of degree ***(n-k-1)*** or less. It is called the "Syndrome polynomial". If ***E(X)*** is the polynomial representing the error pattern caused by the channel, then we have:

$$R(X) = V(X) + E(X) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.22)$$

And it follows as $V(X) = U(X)\,g(X)$, that:

$$E(X) = [A(X) + U(X)]\,g(X) + S(X) \quad (7.23)$$

That is, the syndrome of R(X) is equal to the remainder resulting from dividing the error pattern by the generator polynomial; and the syndrome contains information about the error pattern, which can be used for error correction. Hence syndrome calculation can be accomplished using divider circuits discussed in Sec 7.4, Fig7.5. A " ***Syndrome calculator***" is shown in Fig 7.10.

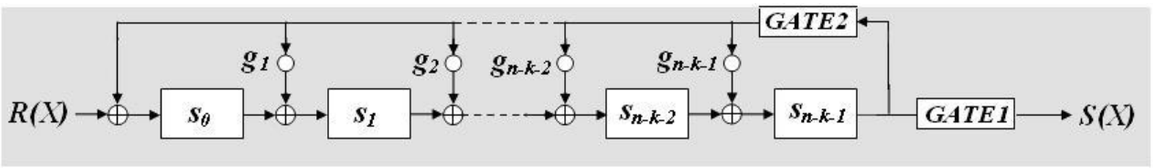


Fig 7.10 Syndrome calculator using (n-k) Shift registers

The syndrome calculations are carried out as below:

1 The register is first initialized. With ***GATE 2 -ON*** and ***GATE1- OFF***, the received vector is entered into the register

2 After the entire received vector is shifted into the register, the contents of the register will be the syndrome, which can be shifted out of the register by turning ***GATE-1 ON*** and ***GATE-2 OFF***. The circuit is ready for processing next received vector.

Cyclic codes are extremely well suited for ***'error detection***' .They can be designed to detect many combinations of likely errors and implementation of error-detecting and error correcting circuits is practical and simple. Error detection can be achieved by employing (or adding) an additional R-S flip-flop to the syndrome calculator. If the syndrome is nonzero, the flip-flop sets and provides an indication of error. Because of the ease of implementation, virtually all error detecting codes are invariably 'cyclic codes'. If we are interested in error correction, then the decoder must be capable of determining the error pattern ***E(X)*** from the syndrome ***S(X)*** and add it to ***R(X)*** to determine the transmitted ***V(X)***. The following scheme shown in Fig 7.11 may be employed for the purpose. The error correction procedure consists of the following steps:

Step1. Received data is shifted into the buffer register and syndrome registers with switches

S_{IN} closed and S_{OUT} open and error correction is performed with S_{IN} open and S_{OUT} closed.

Step2. After the syndrome for the received code word is calculated and placed in the syndrome register, the contents are read into the error detector. The detector is a combinatorial circuit designed to output a ' ***1***' if and only if the syndrome corresponds to a correctable error pattern with an error at the highest order position X^{n-l}. That is, if the detector output is a '***1***' then the received digit at the right most stage of the buffer register is assumed to be in error and will be corrected. If the detector output is '***0***' then the received digit at the right most stage of the buffer is assumed to be correct. Thus the detector output is the estimate error value for the digit coming out of the buffer register.

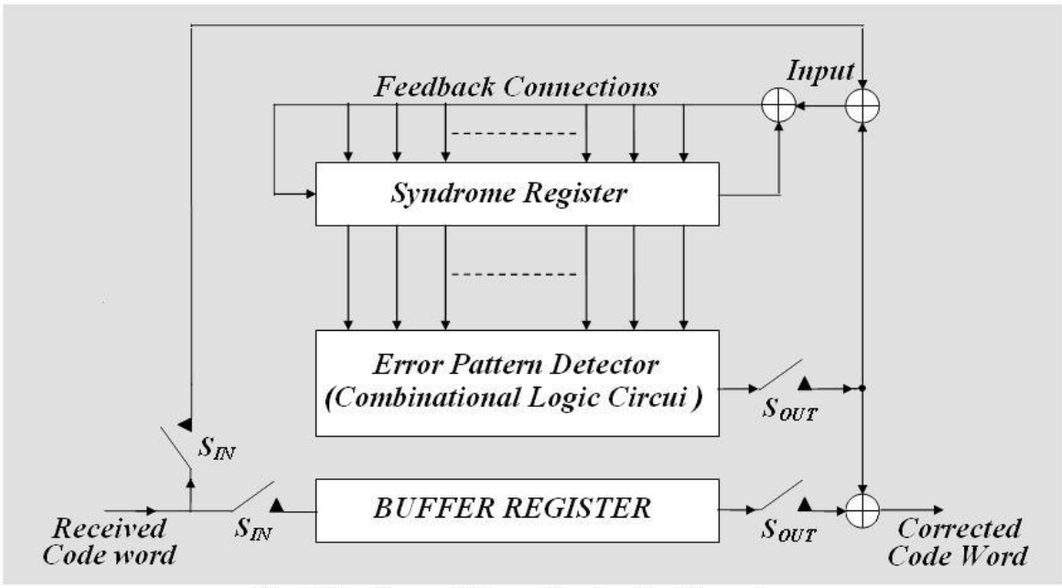


Fig 7.11 General Decoder for Cyclic codes

Step3. In the third step, the first received digit in the syndrome register is shifted right once. If the first received digit is in error, the detector output will be '***1***' which is used for error correction. The output of the detector is also fed to the syndrome register to modify the syndrome. This results in a new syndrome corresponding to the ' ***altered*** 'received code word shifted to the right by one place.

Step4. The new syndrome is now used to check and correct the second received digit, which is now at the right most position, is an erroneous digit. If so, it is corrected, a new syndrome is calculated as in step-3 and the procedure is repeated.

Step5. The decoder operates on the received data digit by digit until the entire received code word is shifted out of the buffer.

At the end of the decoding operation, that is, after the received code word is shifted out of the buffer, all those errors corresponding to correctable error patterns will have been corrected, and the syndrome register will contain all zeros. If the syndrome register does not contain all zeros, this means that an un-correctable error pattern has been detected. The decoding schemes described in Fig 7.10 and Fig7.11 can be used for any cyclic code. However, the practicality depends on the complexity of the combinational logic circuits of the error detector. In fact, there are special classes of cyclic codes for which the decoder can be realized by simpler circuits. However, the price paid for

such simplicity is in the reduction of code efficiency for a given block size.

7.7 Bose- Chaudhury - Hocquenghem (*BCH*) Codes:

One of the major considerations in the design of optimum codes is to make the block size ***n*** smallest for a given size ***k*** of the message block so as to obtain a desirable value of d_{min}. Or for given code length ***n*** and efficiency ***k/n***, one may wish to design codes with largest d_{min}. That means we are on the look out for the codes that have '***best error correcting capabilities***". The ***BCH*** codes, as a class, are one of the most important and powerful error-correcting cyclic codes known. The most common ***BCH*** codes are characterized as follows. Specifically, for any positive integer $m \geq 3$, and $t < 2^m - 1) / 2$, there exists a binary ***BCH*** code (called '***primitive*' *BCH code***) with the following parameters:

$$\textbf{Block length : } n = 2^m - l$$
$$\textbf{Number of message bits : } k \leq n - mt$$
$$\textbf{Minimum distance : } d_{min} \geq 2t + 1$$

Clearly, ***BCH*** codes are "***t - error correcting codes***". They can detect and correct up to' ***t***' random errors per code word. The Hamming ***SEC*** codes can also be described as ***BCH*** codes. The ***BCH*** codes are best known codes among those which have block lengths of a few hundred or less. The major advantage of these codes lies in the flexibility in the choice of code parameters viz: block length and code rate. The parameters of some useful ***BCH*** codes are given below. Also indicated in the table are the generator polynomials for block lengths up to ***31***.

NOTE: Higher order co-efficients of the generator polynomial are at the left. For example, if we are interested in constructing a (***15, 7***) ***BCH*** code from the table we have (***111 010 001***) for the co-efficients of the generator polynomial. Hence

$$g(X) = 1 + X^4 + X^6 + X^7 + X^8$$

n	*k*	*t*	*Generator Polynomial*
7	4	1	1 011
15	11	1	10 011
15	7	2	111 010 001
15	5	3	10 100 110 111
31	26	1	100 101
31	21	2	11 101 101 001
31	16	3	1 000 111 110 101 111
31	11	5	101 100 010 011 011 010 101
31	6	7	11 001 011 011 110 101 000 100 111

For further higher order codes, the reader can refer to Shu Lin and Costello Jr. The alphabet of a ***BCH*** code for $n = (2^m - 1)$ may be represented as the set of elements of an appropriate Galois field, $GF(2^m)$ whose primitive element is α.The generator polynomial of the ***t***-error correcting ***BCH*** code is the least common multiple (***LCM***) of $M_1(X), M_2(X), \ldots M_{2t}(X)$, where $M_i(X)$ is the minimum polynomial of $\alpha^i, i = 1, 2 \ldots 2t$. For further details of the procedure and discussions the reader can refer to J.Das etal.

There are several iterative procedures available for decoding of ***BCH*** codes. Majority of them can be programmed on a general purpose digital computer, which in many practical applications form

an integral part of data communication networks. Clearly, in such systems software implementation of the algorithms has several advantages over hardware implementation

Review questions:

1. Write a standard array for Systematic Cyclic Codes code
2. Explain the properties of binary cyclic codes.
3. With a neat diagrams explain the binary cyclic encoding and decoding
4. Explain how meggit decoder can be used for decoding the cyclic codes.
5. Write short notes on the following
   (iii) BCH codes
6. Draw the general block diagram of encoding circuit using (n-k) bit shift register and explain its operation.
7. Draw the general block diagram of syndrome calculation circuit for cyclic codes and explain its operation.

UNIT – 7

Syllabus:

RS codes, Golay codes, Shortened cyclic codes, Burst error correcting codes. Burst and Random Error correcting codes. **7 Hours**

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996. Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007

Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

UNIT – 7

Cyclic Redundancy Check (CRC) codes:

Cyclic redundancy .check codes are extremely well suited for "error detection". The two important reasons for this statement are, (1) they can be designed to detect many combinations of likely errors. (2) The implementation of both encoding and error detecting circuits is practical. Accordingly, all error detecting codes used in practice, virtually, are of the ***CRC*** -type. In an ***n***-bit received word if a contiguous sequence of 'b-bits' in which the first and the last bits and any number of intermediate bits are received in error, then we say a ***CRC*** "error burst' of length '***b'*** has occurred. Such an error burst may also include an end-shifted version of the contiguous sequence.

In any event, Binary (***n, k***) ***CRC*** codes are capable of detecting the following error patterns:

1. All ***CRC*** error bursts *of* length (***n-k***) or less.
2. A fraction of ($1 - 2^{-(n-k-1)}$) of ***CRC*** error bursts of length ($n - k + 1$).
3. A fraction ($1-2^{-(n-k)}$) of ***CRC*** error bursts of length greater than ($n - k + 1$).
4. All combinations of ($d_{min} - 1$) or fewer errors.
5. All error patterns with an odd number of errors if the generator polynomial ***g (X)*** has an even number of non zero coefficients.

Generator polynomials of three ***CRC*** codes, internationally accepted as standards are listed below. All three contain ***(1 +X)*** as a prime factor. The ***CRC-12*** code is used when the character lengths is ***6-bits***. The others are used for 8-bit characters.

* ***CRC-12 code***: $g(X) = 1 + X + X^2 + X^3 + X^{11} + X^{12}$ *

****CRC-16 code***: $g(X) = 1 + X^2 + X^{15} + X^{16}$

****CRC-CCITT code***: $g(X) = 1 + X^5 + x^{12} + X^{16}$

(Expansion of ***CCITT***: "***Commitè Consultaitif International Tèlèphonique etTèlègraphique***" a Geneva-based organization made up of telephone companies from all over the world)

Maximum Length codes:

For any integer $m \geq 3$, the maximum length codes exist with parameters:

Block length : $n = 2^m - 1$
Message bits : $k = m$
Minimum distance : $d_{min} = 2^{m-1}$

Maximum length codes are generated by polynomials of the form $g(X) = \frac{1 + X^n}{p(X)}$

Maximum length codes are generated by polynomials of degree '***m***'. Notice that any cyclic code generated by a primitive polynomial is a Hamming code of $d_{min} = 3$. It follows then that the maximum length codes are the '***duals***' of Hamming codes. These codes are also referred to as '***pseudo Noise*** (***PN***) ***codes***' or "simplex codes".

Majority Logic Decodable Codes:

These codes form a smaller sub-class of cyclic codes than do the ***BCH*** codes. Their error correcting capabilities, for most interesting values code length and efficiency, are much inferior to ***BCH*** codes. The main advantage is that the decoding can be performed using simple circuits. The concepts are illustrated here with two examples.

Consider a (***7, 3***) simplex code, which is dual to the (***7, 4***) Hamming code. Here $d_{min}=4$ and $t = 1$. This code is generated by ***G*** and corresponding parity check matrix ***H*** given below:

$$G = \begin{matrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{matrix} \qquad H = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

The error vector $e = (e_0, e_1, e_2, e_3, e_4, e_5, e_6)$ is checked by forming the syndromes:

$$s_0 = e_0 + e_4 + e_5; \quad s_2 = e_2 + e_4 + e_5 + e_6;$$

$$s_1 = e_1 + e_5 + e_6; \quad s_3 = e_3 + e_4 + e_6$$

Forming the parity check sum as:

$$A_1 = s_1 = e_1 + e_5 + e_6$$

$$A_2 = s_3 = e_3 + e_4 + e_6$$

$$A_3 = s_0 + s_2 = e_0 + e_2 + e_6$$

It is observed that all the check sums check for the error bit e_6 and no other bit is checked by more than one check sum. Then a majority decision can be taken that $e_6 = 1$ if two or more A_i's are non-zero. If $e_6 = 0$ and any other bit is in error then only one of the A_i's will be non-zero. It is said that the check sums A_i's are orthogonal on the error bit e_6. A circulating ***SR*** memory circuit along with a few logic circuits shown in Fig 7.16 forms the hardware of the decoder.

Initially, the received code vector ***R(X)*** is loaded into the ***SR***'s and check sums A_1, A_2 and A_3 are formed in the circuit. If e_6 is in error then the majority logic output is '***1*** 'and is corrected as it is shifted out of the buffer. If e_6 is correct, then e_5 is checked after one shift of the SR content.

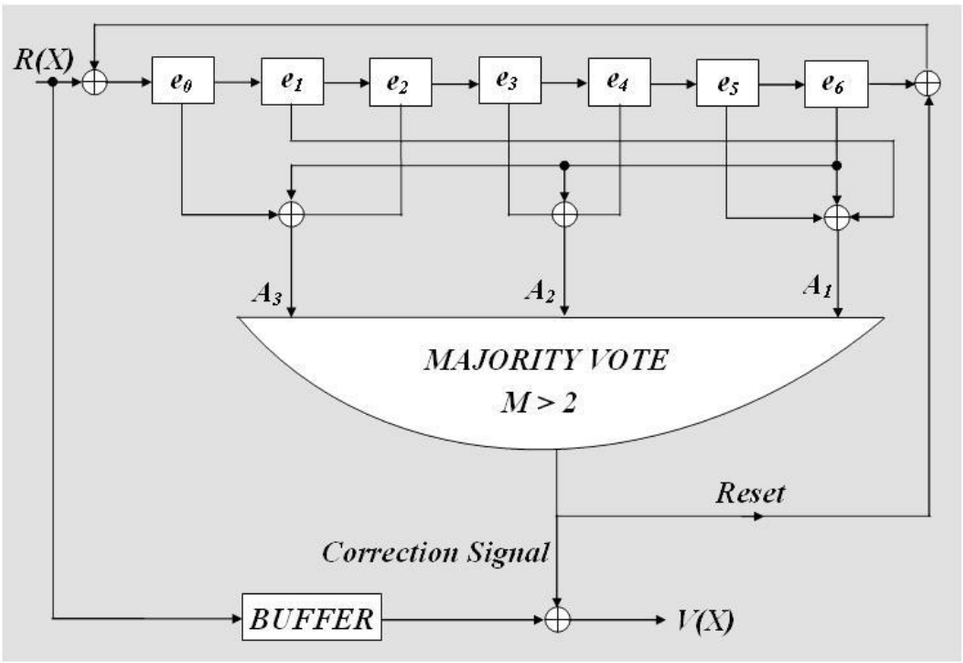


Fig 7.16 A Majority logic decoder for (7,3) Simplex code.

Thus all the bits are checked by successive shifts and the corrected ***V(X)*** is reloaded in the buffer. It is possible to correct single errors only by using two of the check sums. However, by using three check sums, the decoder also corrects some double error patterns. The decoder will correct all single errors and detect all double error patterns if the decision is made on the basis of

(i). $A_1 = 1, A_2 = 1, A_3 = 1$ for single errors (ii).
One or more checks fail for double errors.

We have devised the majority logic decoder assuming it is a Block code. However we should not forget that it is also a cyclic code with a generator polynomial

$$g(X) = 1 + X^2 + X^3 + X^4.$$

Then one could generate the syndromes at the decoder by using a divider circuit as already discussed. An alternative format for the decoder is shown in Fig 7.17. Successive bits are checked for single error in the block. The feed back shown is optional - The feed back will be needed if it is desired to correct some double error patterns.

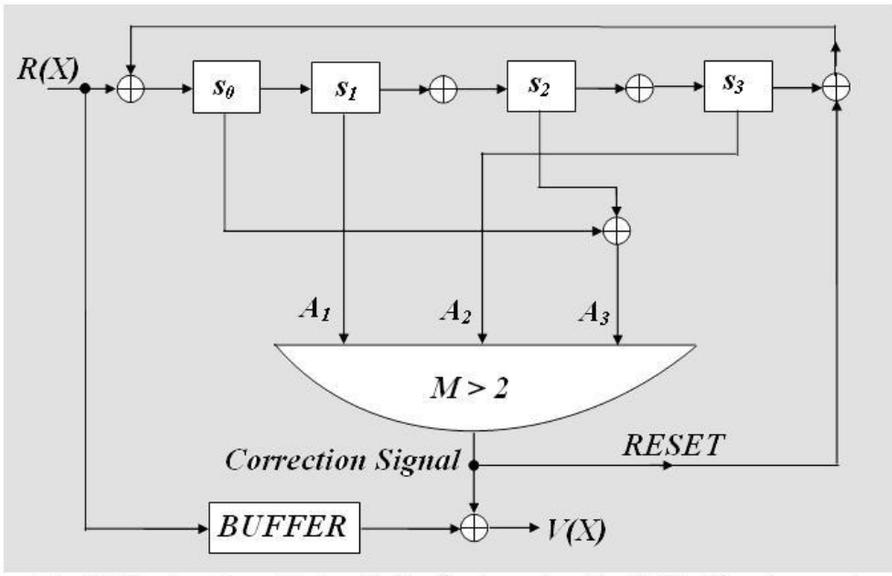


Fig 7.17 Another Majority logic decoder for (7,3) Simplex code.

Let us consider another example – This time the (*7, 4*) Hamming code generated by the polynomial $g(X) = 1 + X + X^3$.

Its parity check matrix is: $H = \begin{matrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & \end{matrix}$

The Syndromes are seen to be

$$s_0 = e_0 + e_3 + (e_5 + e_6)$$
$$s_1 = e_1 + e_3 + e_4 + e_5$$
$$s_2 = e_2 + e_4 + (e_5 + e_6) = e_2 + e_5 + (e_4 + e_6)$$

Check sum $A_1 = (s_0 + s_1) = e_0 + e_1 + (e_4 + e_6)$

It is seen that s_0 and s_2 are orthogonal on $B_1 = (e_5 + e_6)$, as both of them provide check for this sum. Similarly, A_1 and s_2 are orthogonal on $B_2 = (e_4 + e_6)$. Further B_1 and B_2 are orthogonal on e_6. Therefore it is clear that a two-step majority vote will locate the error on e_6.The corresponding decoder is shown in Fig 7.18, where the second level majority logic circuit gives the correction signal and the stored ***R(X)*** is corrected as the bits are read out from the buffer. Correct decoding is achieved if $t < d / 2 = 1$ ***error*** (***d*** = no. of steps of majority vote). The circuit provides majority vote '***1***' when the syndrome state is {*1 0 1*}. The basic principles of both types of decoders, however, are the same. Detailed discussions on the general principles of Majority logic decoding may be found in Shu-Lin and Costello Jr., J.Das etal and other standard books on error control coding. The idea of this section was "only to introduce the reader to the concept of majority logic decoding.

The Hamming codes ($2^m-1, 2^m-m-1$), ***m*** any integer, are majority logic decodable, (*l5, 7*) ***BCH*** code with $t \leq 2$ is *1*-step majority logic decodable. Reed-Muller codes, maximum length (simplex) codes Difference set codes and a sub-class of convolutional codes are examples majority logic decodable codes.

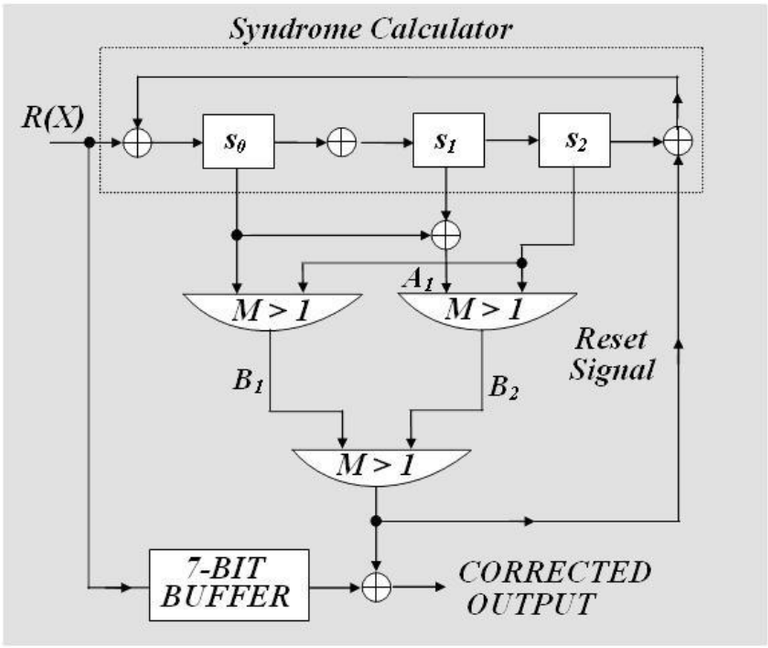


Fig 7.18 Two-Step Majority decoder for (7,4) Hamming code.

Shortened cyclic codes:

The generator polynomials for the cyclic codes, in general, are determined from among the divisors of $X^n + 1$. Since for a given ***n*** and ***k***, there are relatively few divisors, there are usually very few cyclic codes of a given length. To overcome this difficulty and to increase the number of pairs (***n, k***) for which useful codes can be constructed, cyclic codes are often used in shortened form. In this form the last ' ***j'*** information digits are always taken to be zeros and these are not transmitted. The decoder for the original cyclic code can decode the shortened cyclic codes simply by padding the received (***n-j***) tuples with '***j***' zeros. Hence, we can always construct an (***n-j, k-j***) shortened cyclic code starting from a (***n, k***) cyclic code. Therefore the code thus devised is a sub-set of the cyclic code from which it was derived - which means its minimum distance and error correction capability is at least as great as that of the original code. The encoding operation, syndrome calculation and error correction procedures for shortened codes are identical to those described for cyclic codes. This implies- shortened cyclic codes inherit nearly all of the implementation advantages and much of the mathematical structure of cyclic codes.

Golay codes:

Golay code is a (***23, 12***) perfect binary code that is capable of correcting any combination of three or fewer random errors in a block of ***23*** bits. It is a perfect code because it satisfies the Hamming bound with the equality sign for $t = 3$ as:

$$\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{23-12} = 2^{23}$$

The code has been used in many practical systems. The generator polynomial for the code is obtained from the relation $(X^{23}+1) = (X+1)\, g_1(X)\, g_2(X)$, where:

$g_1(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}$ and $g_2(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}$

The encoder can be implemented using shift registers using either $g_1(X)$ or $g_2(X)$ as the divider polynomial. The code has a minimum distance, $d_{min} = 7$. The extended Golay code, a (*924, 12*) code has $d_{min} = 8$. Besides the binary Golay code, there is also a perfect ternary *(11, 6)* Golay code with $d_{min} = 5$.

Reed-Solomon Codes:

The Reed-Solomon (***RS***) codes are an important sub class of ***BCH*** codes, where the symbols are from ***GF (q), q ≠ 2^m*** in general, but usually taken as 2^m. The encoder for an RS code differs from a binary encoder in that it operates on multiple bits rather than individual bits. A' ***t***'-error correcting ***RS*** code has the following parameters.

Block length: n = (q - 1) symbols

Number of parity Check symbols: r = (n - k) = 2t

Minimum distance: dmin = (2t + 1)

The encoder for an ***RS*** (***n, k***) code on ***m***-bit symbols groups the incoming binary data stream into blocks, each ***km*** bits long. Each block is treated as ***k*** symbols, with each symbol having ***m***-bits. The encoding algorithm expands a block of ***k*** symbols by adding (***n - k***) redundant symbols. When m is an integer power of ***2***, the ***m*** - bit symbols are called 'Bytes'. A popular value of ***m*** is ***8*** and ***8***-bit ***RS*** codes are extremely powerful. Notice that no (***n, k***) linear block code can have $d_{min} > (n - k + 1)$. For the RS code the block length is one less than the size of a code symbol and minimum distance is one greater than the number of parity symbols - "*The* d_{min} *is always equal to the design distance of the code*". An (***n, k***) linear block code for which $d_{min} = (n-k-l)$ is called 'Maximum - distance separable' code. Accordingly, every ***RS*** code is ' *maximum - distance separable*' code-They make highly efficient use of redundancy and can be adjusted to accommodate wide range of message sizes. They provide wide range of code rates (***k / n***) that can be chosen to optimize performance. Further, efficient decoding techniques are available for use with ***RS*** codes (usually similar to those of ***BCH*** codes).

Reed-Muller Codes (***RM*** codes) are a class of binary group codes, which are majority logic decodable and have a wide range of rates and minimum distances. They are generated from the Hadamard matrices. (Refer J. Das etal).

CODING FOR BURST ERROR CORRECTION

The coding and decoding schemes discussed so far are designed to combat random or independent errors. We have assumed, in other words, the channel to be "Memory less". However, practical channels have 'memory' and hence exhibit mutually d ependent signal transmission impairments. In a 'fading channel', such impairment is felt, particul arly when the fading varies slowly compared to one symbol duration. The 'multi-path' impairment involv es signal arrivals at the receiver over two or more paths of different lengths with the effect that the signals "arrive out of phase" with each other and the cumulative received signal are distorted. High-Frequency (HF) and troposphere propagation in radio channels suffer from such a phenomenon. Further , some channels suffer from switching noise and other burst noise (Example: Telephone channels or channels disturbed by pulse jamming impulse noise in the communication channel causes transmission errors to cluster into 'bursts'). All of these time-correlated impairments results in statistical dependence among successive symbol transmissions. The disturbances tend to cause errors that occur in bursts rather than isolated events.

Once the channel is assumed to have memory, the errors that occur can no longer be characterized as single randomly distributed errors whose occurrence is independent from bit to bit. Majority of the codes: Block, Cyclic or Convolutional codes are designed to combat such random or independent errors. They are, in general, not efficient for correcting burst errors. The channel memory causes degradation in the error performance.

Many coding schemes have been proposed for channels with memory. Greatest problem faced is the difficulty in obtaining accurate models of the frequently time-varying statistics of such channels. We shall briefly discuss some of the basic ideas regarding such codes. (A detailed discussion of burst error correcting codes is beyond the scope of this book). We start with the definition of burst length, ' ***b***' and requirements on a (***n, k***) code to correct error burst. "An error burst of length ' ***b***' is defined as a sequence of error symbols confined to ' ***b***' consecutive bit positions in which the first and the last bits are non-zero"

For example, an error vector (***00101011001100***) is a burst of ***b = 10***. The error vector (***001000110100***) is a burst of ***b = 8.*** A code that is capable of correcting all burst errors of length ' ***b***' or less is called a " ***b-burst-error-correcting code***". Or the code is said to have a ***burst error correcting capability*** = ***b***. Usually for proper decoding, the ***b***-symbol bursts are separated by a guard space of ' ***g***' symbols. Let us confine, for the present, ourselves for the construction of an (***n***, ***k***) code for a given ***n*** and ***b*** with as small a redundancy (***n - k***) as possible. Then one can make the following observations.

Start with a code vector ***V*** with an error burst of length 2***b*** or less. This code vector then may be expressed as a linear combination (vector sum) of the vectors V_1 and V_2 of length ***b*** or less. Therefore in the standard array of the code both V_1 and V_2 must be in the same co-set. Further, if one of these is assumed to be the co-set leader (i.e. a correctable error pattern), then the other vector which is in the same co-set turns out to be an un-correctable error pattern. Hence, this code will not be able to correct all error bursts of length ***b*** or less. Thus we have established the following assertion:

Assertion-1: ***"A necessary condition for a (n ,k) linear code to b e able to correct all error bursts of length b or less is that no error burst of length 2b or less be a code vector".***

Next let us investigate the structure of code vectors whose non zero components are confined to the first ' ***b***' bits. There are, clearly, 2^b such code vectors. No two such vectors can be in the same co-set of the standard array; otherwise their sum, which is again a burst of ***b*** or less, would be a code vector. Therefore these 2^b vectors must be in 2^b distinct co-sets. For an ***(n, k)*** code we know that there are $2^{(n-k)}$ co-sets. This means that ***(n-k)*** must be at least equal to ' ***b***'. Thus we have established another important assertion.

Assertion-2: ***"The number of parity check bits of an (n, k) linear code that has no bursts of length b or less as a code vector is at least 'b', i.e. (n -k) ≥ b***

Combining the two assertions, now we can conclude that: "The number of parity check bits of a b-burst error correcting code must be at least 2b"

i. e. $(n-k) \geq 2b$ ……. (9.1)

From Eq. (9.1) it follows that the burst-error-correcting capability of an (***n***, ***k***) code is at most ***(n-k)//2***. That is, the upper bound on the burst-error-correcting capability of an (***n***, ***k***) linear code is governed by:

$$b \geq (n-k)/2 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (9.2)$$

This bound is known by the name " ***Reiger Bound***" and it is used to define the ***burst correcting efficiency***, *z*, of an (***n, k***) codes as

$$z = 2b/(n-k) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (9.3)$$

Whereas most useful random error correcting codes have been devised using analytical techniques, for the reasons mentioned at the beginning of this section, the best burst- error correcting codes have to be found through computer aided search procedures. A short list of high rate burst-error- correcting cyclic codes found by computer search is listed in Table-9.1.

If the code is needed for 'detecting' error bursts of length' ***b***', then the number of check bits must satisfy:

$$(n-k) \geq b \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (9.4)$$

Some of the famous block/cyclic and convolution codes designed for correcting burst errors are Burton, Fire, R-S, Berlekemp - Preparata-Massey , Iwadare and Adaptive Gallager codes. Of these Fire codes have been extensively used in practice. A detailed discussion on these codes is available in Shu-Lin et-all and J.Das et-all. (Refer – Bibliogr aphy)

Burst and Random Error Correcting Codes:

In most practical systems, error occurs neither independently, at random, nor in well-defined bursts. As a consequence codes designed for random error correction or single-burst-error correction will become either inefficient or in adequate for tackling a mixture of random and burst errors. For channels in which both types of error occur, it is better to design codes that can correct both types of errors. One technique, which only requires knowledge of the duration or span of channel memory, not its exact statistical characterization, is the use of " ***Time diversity or Interleaving***".

Interleaving the code vectors before transmission and de-interleaving after reception causes the burst errors to be spread out in time and the decoder can handle them as if they were random errors. Since, in all practical cases, the channel memory decreases with time separation, the idea behind interleaving is only to separate the code word symbols in time. The interleaving times are similarly filled by symbols of other code words. " *Separating the symbols in time effectively transforms a channel with memory to a 'memory less' channel* ", and there by enable the random error correcting codes to be useful in a bursty-noise channel.

The function of an interleaver is to shuffle the code symbol over a span of several block lengths (for block codes) or several constraint lengths (for convolutional codes). The span needed is usually determined from the knowledge of the burst length. Further the details of the bit-redistribution pattern must be known at the receiver end to facilitate de-interleaving and decoding. Fig 9.1 illustrates the concept of interleaving.

The un-interleaved code words shown in Fig 9.1(*a*) are assumed to have a single error correcting capability with in each six-symbol sequence. If the memory span of the channel is one code word in duration, a six symbol time noise burst could destroy the information contained in one or two code words. On the contrary, suppose encoded data were interleaved as shown in Fig 9.1(*b*), such that each code symbol of each code word is separated from its pre-interleaved neighbors by a span of six symbol times. The result of an error burst as marked in Fig.9.1 is to affect one code symbol from each of the original six code words. Upon reception, the stream is de-interleaved and decoded as

though a single-random error has occurred in each code word. Clearly the burst noise has no degrading effect on the final sequence.

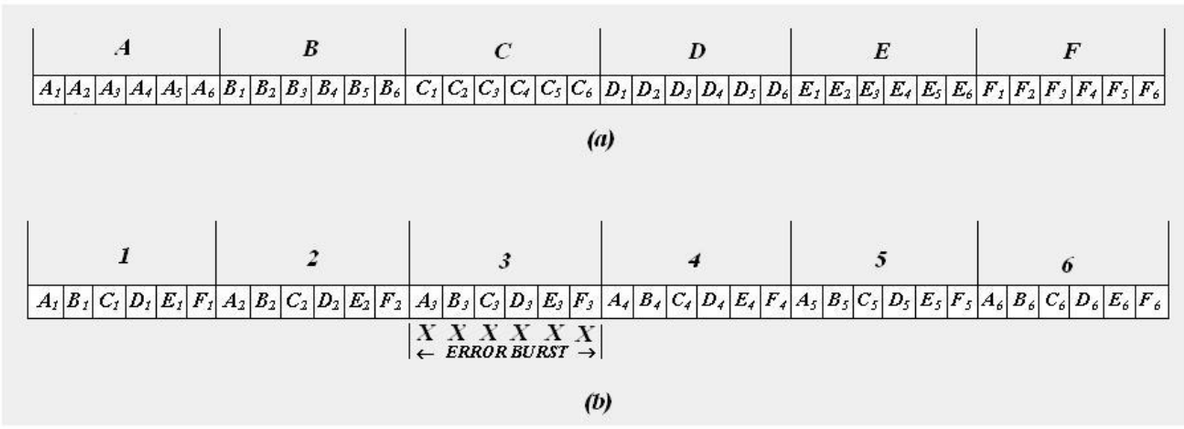


Fig 9.1 Illustrating Interleaving (a) Original Un-Interleaved code words made up of six code symbols. (b) Interleaved code symbols.

Block Interleaving:

Given an (n, k) cyclic code, it is possible to construct a (λn, λk) cyclic " ***interlaced code***" by simply arranging λ code vectors of the original code into λ rows of a rectangular array and then transmitting them column by column. The parameter ' λ' is called " ***Degree of interlacing***". By such an arrangement, a burst of length λ or less will affect no more than one symbol in each row since transmission is done on a column by column basis. If the original code (whose code words are the rows of the ($\lambda n \times \lambda k$) matrix) can correct single errors, then the interlaced code can correct single bursts of length λ or less. On the other hand if the original code has an error correcting capability of'' t', t >1, then the interlaced code is capable of correcting any combination of t-error bursts of length λ or less. The performance of the (λn, λk) interleaved cyclic code against purely random errors is identical to that of the original (n, k) cyclic code from which it was generated.

The block interleaver accepts the coded symbols in blocks from the encoder, permutes the symbols and then feeds the re-arranged symbols to the modulator. The usual permutation of block is accomplished by 'filling the rows' of a ' λ' row by ' n'- column 'array' with the encoded sequence. After the array is completely filled, the symbols are then fed to the modulator ' *one column at a time*' and transmitted over the channel. At the receiver the code words are re-assembled in a complimentary manner. The de-inter leaver accepts the symbols from the de-modulator, de-interleaves them and feeds them to the decoder - symbols are entered into the de interleaver array by columns and removed by rows. The most important characteristics of block interleaver may be summarized as follows:

1) Any burst of length less than λ contiguous channel symbol errors result in isolated errors at the de-interleaver output that are separated from each other by at least n-symbols.

2) Any $q.\lambda$ burst of errors, where q >1, results in output bursts from the de-interleaver of not more than q – symbol errors. Each output burst is separated fr om the other burst by not less than n-q symbols. The notation q means the smallest integer not less than q and q means the largest integer not greater than q.

3) A periodic sequence of single errors spaced λ- symbols apart results in a single burst of errors of length ' n' at the de-interleaver output.

4) The interleaver/ de-interleaver end – to –end delay is approximately $2\lambda n$ symbol time units to be filled at the receiver before decoding begins. Therefore, the minimum end- to- end delay is ($2\lambda n-2n+2$) symbol time units. This does not include any channel propagation delay.

5) The memory requirement, clearly, is λn symbols for each location (interleaver and de-interleaver). However, since the $\lambda \times n$ array needs to be (mostly) filled before it can be read out, a memory of $2\lambda n$ symbols is generally implemented at each location to allow the emptying of one $\lambda \times n$ array while the other is being filled , and vice versa.

Finally a note about the simplest possible implementation aspect- If the original code is cyclic then the interleaved code is also cyclic. If the original code has a generator polynomial $g(X)$, the interleaved code will have the generator polynomial $g(X^{\lambda})$. Hence encoding and decoding can be done using shift registers as was done for cyclic codes. The modification at the decoder for the interleaved codes is done by replacing each shift register stage of the original decoder by λ – stages without changing other connections. This modification now allows the decoder to look at successive rows of the code array on successive decoder cycles. It then follows that if the decoder for the original cyclic code is simple so will it be for the interleaved code. "Interleaving technique is inde ed an effective tool for deriving long powerful codes from short optimal codes".

Example 9.1:

Let us consider an interleaver with $n = 4$ and $\lambda = 6$. The corresponding (6×4) array is shown in Fig. 9.2(*a*). The symbols are numbered indicating the sequence of transmission. In Fig 9.2(*b*) is shown an error burst of five-symbol time units-The symbols shown encircled suffer transmission errors. After de- interleaving at the receiver, the sequence is:

← One Code word →																			
1	7	13	(19)	2	8	14	(20)	3	9	15	21	4	10	(16)	22	5	11	(17)	23	6	12	(18)	24

Observe that in the de-interleaved sequence, each code word does not have more than one error. The smallest separation between symbols in error is $n = 4$.

Next, with $q = 1.5$, $q\lambda = 9$. Fig 9.2(*c*) illustrates an example of 9-symbol error burst. After de-interleaving at the receiver, the sequence is:

← One Code word →																			
1	7	(13)	19	2	8	(14)	20	3	9	(15)	21	4	(10)	(16)	22	5	(11)	(17)	23	6	(12)	(18)	24

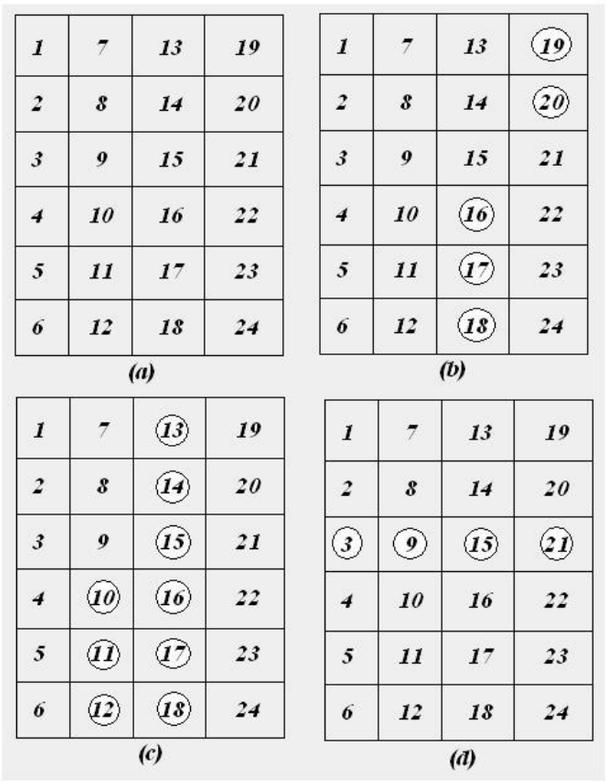


Fig 9.2 Example of a Block Interleaver
(a) A $\lambda \times n = 6 \times 4$ Interleaver
(b) A Five symbol Error Burst
(c) Nine symbol Error Burst
(d) Periodic signal error spaced $\lambda = 6$ symbols apart.

Each Row is a 4 bit Code word.

The encircled symbols are in error. It is seen that the bursts consists of no more than $1.5 = 2$ contiguous symbols per code word and they are separated by at least $n - 1.5 = 4 - 1 = 3$ symbols. Fig.9.2 (*d*) illustrates a sequence of single errors spaced by $\lambda = 6$ symbols apart. After de-interleaving at the receiver, the sequence is:

← One Code word →																			
1	7	13	19	2	8	14	20	3	9	15	21	4	10	16	22	5	11	17	23	6	12	18	24

It is seen that de-interleaved sequence has a single error burst of length $n = 4$ symbols. The minimum end –to –end delay due to interleaver and de-interl eaver is $(2\lambda n - 2n + 2) = 42$ symbol time units. Storage of $\lambda n = 24$ symbols required at each end of the channel. As said earlier, storage for $2\lambda n = 48$ symbols would generally be implemented.

Example 9.2: Interleaver for a BCH code.

Consider a $(15, 7)$ ***BCH*** code generated by $g(X) = 1 + X + X^2 + X^4 + X^8$. For this code $d_{min} = 5$, $t = \frac{d_{mim} - 1}{2} = 2$ .With $\lambda = 5$, we can construct a $(75, 35)$ interleaved code with a burst error correcting capability of $b = \lambda t = 10$. The arrangement of code words, similar to Example 9.1, is shown in Fig 9.3. A 35-bit message block is divided into five 7-bit message blocks and five code words of length 15 are generated using $g(X)$.These code words are arranged as 5-rows of a 5×15 matrix. The columns of the matrix are transmitted in the sequence shown as a 75-bit long code vector.

← Each rows is 15 – bit code word →

1	6	11	….	31	(36)	…..	(66)	71
2	7	12	….	(32)	(37)	…..	67	72
3	8	13	….	(33)	(38)	…..	68	73

4	*(9)*	*14*	*….*	*(34)*	*39*	*…..*	*69*	*74*
5	*10*	*15*	*….*	*(35)*	*40*	*.....*	*(70)*	*75*

Fig 9.3 Block Interleaver for a (15, 7) BCH code.

To illustrate the burst and random error correcting capabilities of this code, we have put the bit positions ***9***, ***3***2 to ***38***, ***66*** and ***70*** in parenthesis, indicating errors occurred in these positions .The de-interleaver now feeds the rows of Fig 9.3 to the decoder. Clearly each row has a maximum of two errors and since the (***15, 7***) ***BCH*** code, from which the rows were constructed, is capable of correcting up to two errors per row. Hence the error pattern shown in parenthesis in the Figure can be corrected. The isolated errors in bit positions ***9***, ***66*** and ***70*** may be thought of as random errors while the cluster of errors in bit positions ***32*** to ***38*** as a burst error.

Convolutional Interleaving:

Convolution interleavers are somewhat simpler and more effective compared to block interleavers. A (***b*** × ***n***) periodic (convolutional) interleaver is shown in Fig 9.4.The code symbols are shifted sequentially into the bank of ***n***-shift registers. Each successive register introduces a delay ' ***b***'. i.e., the successive symbols of a codeword are delayed by {***0***, ***b***, 2***b*** … (***n-1***) ***b***} symbol units respectively. Because of this, the symbols of one codeword are placed at distances of ***b***-symbol units in the channel stream and a burst of length ' ***b***' separated by a guard space of (***n***-1)***b*** symbol units only affect one symbol per codeword. In the receiver, the code words are reassembled through complementary delay units and decoded to correct single errors so generated. If the burst length ***l*** > ***b*** but ***l*** ≤ ***2b***, then the (***n***, ***k***) code should be capable of correcting two errors per code words. To economize on the number of SR's as shown in Fig 9.4 and clock them at a period of ***nTo***, where ***To*** = symbol duration. This would then ensure the required delays of {***b***, 2***b***… (***n***-1) ***b***} symbol units.

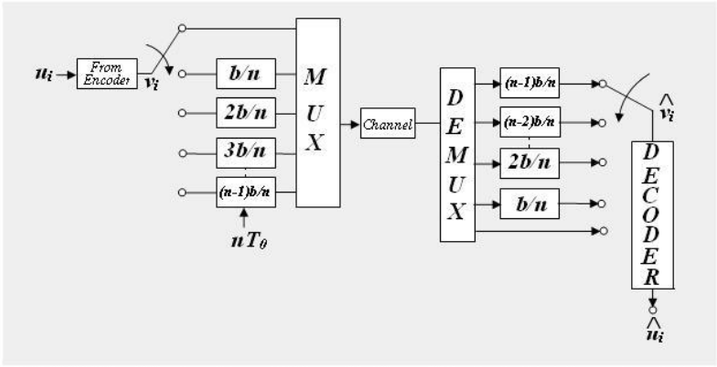


Fig 9.4 Shift Register Implementation of a (b × n) Periodic Interleaver/De-InterLeaver
T_θ = Symbol duration

As illustrated in Example 9.2, if one uses a (***15, 7***) ***BCH*** code with ***t***=2, then a burst of length ≤ 2***b*** can be corrected with a guard space of (***n***-1) ***b*** = ***14b***. This ***14*** – to – ***2*** –guard space to burst length ratio is too large, and hence codes with smaller values of ***n*** are preferable. Convolutional codes with interleaver may also be used. The important advantage of convolutional interleaver over block interleaver is that, with convolutional interleaving the end-to-end delay is (***n***-1) ***b*** symbol units and the memory required at both ends of the channel is ***b*** (***n***-1)/***2.*** This means, there is a reduction of one half in delay and memory over the block interleaving requirements.

Review Questions:

1. What are RS codes? How are they formed?
2. Write down the parameters of RS codes and explain those parameters with an example.
3. List the applications of RS codes.
4. Explain why golay code is called as perfect code.
5. Explain the concept of shortened cyclic code.
6. What are burst error controlling codes?
7. Explain clearly the interlacing technique with a suitable example.
8. What are Cyclic Redundancy Check (CRC) codes

UNIT – 8: CONVOLUTIONAL CODES

Syllabus:
Convolution Codes, Time domain approach. Transform domain approach. **7 Hours**

Text Books:

Digital and analog communication systems, K. Sam Shanmugam, John Wiley, 1996. Digital communication, Simon Haykin, John Wiley, 2003.

Reference Books:

ITC and Cryptography, Ranjan Bose, TMH, II edition, 2007
Digital Communications - Glover and Grant; Pearson Ed. 2nd Ed 2008

Unit 8

CONVOLUTIONAL CODES

In block codes, a block of ***n***-digits generated by the encoder depends only on the block of ***k***-data digits in a particular time unit. These codes can be generated by combinatorial logic circuits. In a convolutional code the block of ***n***-digits generated by the encoder in a time unit depends on not only on the block of ***k***-data digits with in that time unit, but also on the preceding ' ***m***' input blocks. An (***n, k, m***) convolutional code can be implemented with ***k***-input, ***n***-output sequential circuit with input memory ***m***. Generally, ***k*** and ***n*** are small integers with ***k*** < ***n*** but the memory order ***m*** must be made large to achieve low error probabilities. In the important special case when ***k = 1***, the information sequence is not divided into blocks but can be processed continuously.

Similar to block codes, convolutional codes can be designed to either detect or correct errors. However, since the data are usually re-transmitted in blocks, block codes are better suited for error detection and convolutional codes are mainly used for error correction.

Convolutional codes were first introduced by Elias in 1955 as an alternative to block codes. This was followed later by Wozen Craft, Massey, Fano, Viterbi, Omura and others. A detailed discussion and survey of the application of convolutional codes to practical communication channels can be found in Shu-Lin & Costello Jr., J. Das etal and other standard books on error control coding.

To facilitate easy understanding we follow the popular methods of representing convolutional encoders starting with a connection pictorial - needed for all descriptions followed by connection vectors.

8.1 Connection Pictorial Representation:

The encoder for a (rate ***1/2***, ***K = 3***) or (***2, 1, 2***) convolutional code is shown in Fig.8.1. Both sketches shown are one and the same. While in Fig.8.1 (a) we have shown a 3-bit register, by noting that the content of the third stage is simply the output of the second stage, the circuit is modified using only two shift register stages. This modification, then, clearly tells us that" the memory requirement ***m = 2***. For every bit inputted the encoder produces two bits at its output. Thus the encoder is labeled (n, k, m)→ (***2, 1, 2***) encoder.

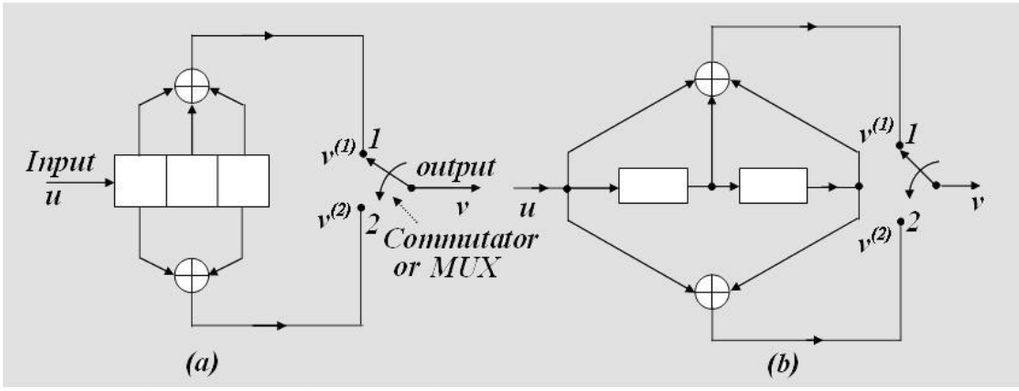


Fig 8.1 A (2, 1, 2) Encoder (a) Representation using 3-bit shift register.
(b)Equivalent representation requires only two shift register stages.

At each input bit time one bit is shifted into the left most stage and the bits that were present in the

registers shifted to the right by one position. Output switch (commutator /MUX) samples the output of each X-OR gate and forms the code symbol pairs for the bits introduced. The final code is obtained after flushing the encoder with "***m***" zero's where '***m***'- is the memory order (In Fig.8.1, ***m = 2***). The sequence of operations performed by the encoder of Fig.8.1 for an input sequence ***u = (101)*** are illustrated diagrammatically in Fig 8.2.

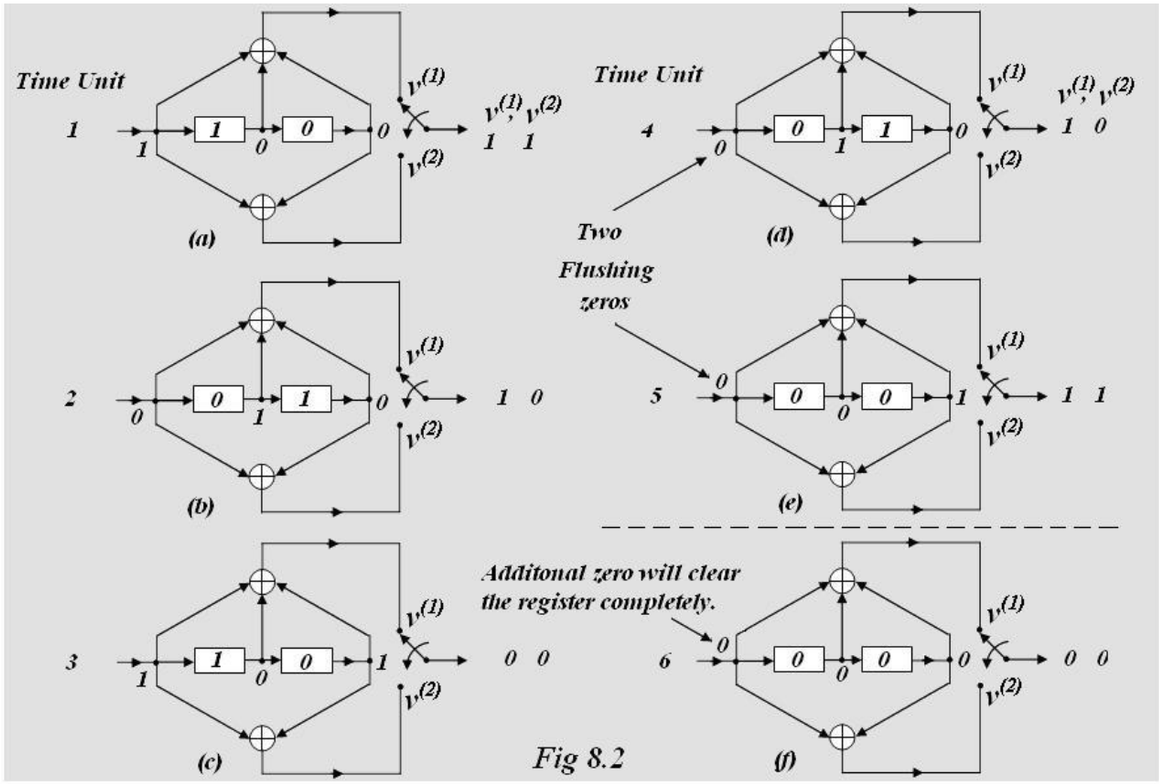


Fig 8.2

From Fig 8.2, the encoding procedure can be understood clearly. Initially the registers are in Re-set mode i.e. (***0, 0***). At the first time unit the input bit is ***1***. This bit enters the first register and pushes out its previous content namely ' ***0***' as shown, which will now enter the second register and pushes out its previous content. All these bits as indicated are passed on to the X-OR gates and the output pair (***1, 1***) is obtained. The same steps are repeated until time unit ***4***, where zeros are introduced to clear the register contents producing two more output pairs. At time unit ***6***, if an additional ' ***0***' is introduced the encoder is re-set and the output pair (***0, 0***) obtained. However, this step is not absolutely necessary as the next bit, whatever it is, will flush out the content of the second register. The ' ***0***' and the ' ***1***' indicated at the output of second register at time unit ***5*** now vanishes. Hence after (***L+m***) = 3 + 2 = 5 time units, the output sequence will read ***v*** = (***11, 10, 00, 10, 11***). (Note: ***L*** = length of the input sequence). This then is the code word produced by the encoder. It is very important to remember that " ***Left most symbols represent earliest transmission***".

As already mentioned the convolutional codes are intended for the purpose of error correction. However, it suffers from the ' *problem of choosing connections*' to yield good distance properties. The selection of connections indeed is very complicated and has not been solved yet. Still, good codes have been developed by computer search techniques for all ***constraint lengths*** less than ***20***. Another point to be noted is that the convolutional codes do not have any particular block size. They can be periodically truncated. Only thing is that they require ***m***-zeros to be appended to the end of the input sequence for the purpose of ' ***clearing***' or ' ***flushing***' or ' ***re-setting***' of the encoding

shift registers off the data bits. These added zeros carry no information but have the effect of reducing the code rate below (***k/n***). To keep the code rate close to (***k/n***), the truncation period is generally made as long as practical.

The encoding procedure as depicted pictorially in Fig 8.2 is rather tedious. We can approach the encoder in terms of "Impulse response" or "gene rator sequence" which merely represents the response of the encoder to a single ' ***1***' bit that moves through it.

8.2 Convolutional Encoding – Time domain approach:

The encoder for a (***2, 1, 3***) code is shown in Fig. 8.3. Here the encoder consists of ***m=3*** stage shift register, ***n=2*** modulo-2 adders (X-OR gates) and a multiplexer for serializing the encoder outputs. Notice that module-2 addition is a linear operation and it follows that all convolution encoders can be implemented using a " ***linear feed forward shift register circuit***".

The "information sequence' $u = (u_1, u_2, u_3 \ldots\ldots)$ enters the encoder one bit at a time starting from u_1. As the name implies, a convolutional encoder operates by performing convolutions on the information sequence. Specifically, the encoder output sequences, in this case $v^{(1)} = \{v_1^{(1)}, v_2^{(1)}, v_3^{(1)} \ldots\}$and $v^{(2)} = \{v_1^{(2)}, v_2^{(2)}, v_3^{(2)} \ldots\}$ are obtained by the discrete convolution of the information sequence with the encoder "impulse responses'. The impulse responses are obtained by determining the output sequences of the encoder produced by the input sequence $u = (1, 0, 0, 0\ldots)$ .The impulse responses so defined are called ***'generator sequences***' of the code. Since the encoder has a ***m***-time unit memory the impulse responses can last at most $(m+1)$ time units (That is a total of $(m+1)$ shifts are necessary for a message bit to enter the shift register and finally come out) and are written as:

$$g^{(i)} = \{g_1^{(i)}, g_2^{(i)}, g_3^{(i)} \ldots g_{m+1}^{(i)}\}.$$

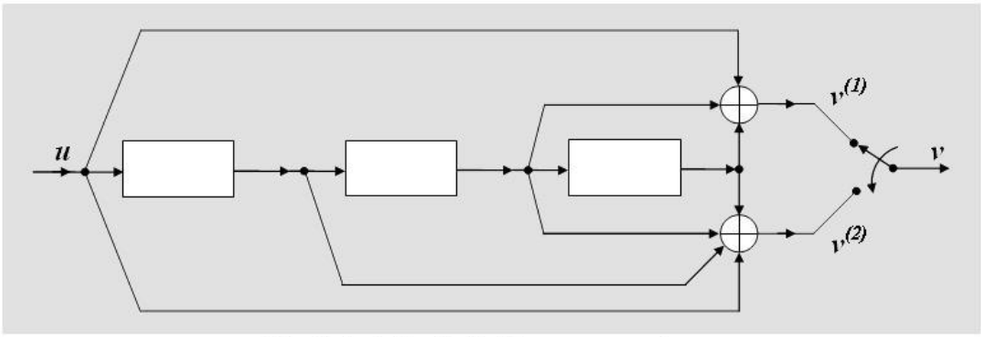


Fig 8.3 A (2, 1, 3) binary encoder.

For the encoder of Fig.8.3, we require the two impulse responses,

$g^{(1)} = \{g_1^{(1)}, g_2^{(1)}, g_3^{(1)}, g_4^{(1)}\}$ and

$g^{(2)} = \{g_1^{(2)}, g_2^{(2)}, g_3^{(2)}, g_4^{(2)}\}$

By inspection, these can be written as: $g^{(1)} = \{1, 0, 1, 1\}$ and $g^{(2)} = \{1, 1, 1, 1\}$

Observe that the generator sequences represented here is simply the 'connection vectors' of the encoder. In the sequences a '***1***' indicates a connection and a '***0***' indicates no connection to the corresponding X - OR gate. If we group the elements of the generator sequences so found in to pairs,

we get the overall impulse response of the encoder, Thus for the encoder of Fig 8.3, the 'over-all impulse response' will be:

$$v = (11, 01, 11, 11)$$

The encoder outputs are defined by the convolution sums:

$$v^{(1)} = u * g^{(1)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (8.1\ a)$$

$$v^{(2)} = u * g^{(2)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (8.1\ b)$$

Where * denotes the 'discrete convolution', which i mplies:

$$v_l^{(j)} = \sum_{i=0}^{m} u_{l-i} \cdot g_{i+1}^{(j)}$$

$$= u_l g_1^{(j)} + u_{l-1}\ g_2^{(j)} + u_{l-2}\ g_3^{(j)} + \ldots\ldots + u_{l-m}\ g_{m+1}^{(j)} \quad \ldots\ldots\ldots\ldots \quad (8.2)$$

for $j = 1, 2$ and where $u_{l-i} = 0$ for all $l < i$ and all operations are modulo - 2. Hence for the encoder of Fig 8.3, we have:

$$v_l^{(1)} = u_l + u_{l-2} + u_{l-3}$$

$$v_l^{(2)} = u_l + u_{l-1} + u_{l-2} + u_{l-3}$$

This can be easily verified by direct inspection of the encoding circuit. After encoding, the two output sequences are multiplexed into a single sequence, called the "***code word***" for transmission over the channel. The code word is given by:

$$v = \{v_1^{(1)}\ v_1^{(2)}, v_2^{(1)}\ v_2^{(2)}, v_3^{(1)}\ v_3^{(2)} \ldots\}$$

Example 8.1:

Suppose the information sequence be $u = (10111)$. Then the output sequences are:

$$v^{(1)} = (1\ 0\ 1\ 1\ 1) * (10\ 1\ 1)$$
$$= (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1),$$

$$v^{(2)} = (1\ 0\ 1\ 1\ 1) * (1\ 1\ 1)$$
$$= (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1),$$

and the code word is

$$v = (11, 01, 00, 01, 01, 01, 00, 11)$$

The discrete convolution operation described in Eq (8.2) is merely the addition of shifted impulses. Thus to obtain the encoder output we need only to shift the overall impulse response by '***one branch word***', multiply by the corresponding input sequence and then add them. This is illustrated in the table below:

INPUT	*OUT PUT*
1	*1 1 0 1 1 1 1*
0	*0 0 0 0 0 0 0 0 -----one branch word shifted sequence*
1	*1 1 0 1 1 1 1 ---Two branch word shifted*
1	*1 1 0 1 1 1 1 1*
1	*1 1 0 1 1 1 1 1*
Modulo -2 sum	*1 1 0 1 0 0 0 1 0 1 0 1 0 0 1 1*

The Modulo-2 sum represents the same sequence as obtained before. There is no confusion at all with respect to indices and suffices! Very easy approach - super position or linear addition of shifted impulse response - demonstrates that the convolutional codes are linear codes just as the block codes and cyclic codes. This approach then permits us to define a *'Generator matrix*' for the convolutional encoder. Remember, that interlacing of the generator sequences gives the overall impulse response and hence they are used as the rows of the matrix. The number of rows equals the number of information digits. Therefore the matrix that results would be " ***Semi-Infinite***". The second and subsequent rows of the matrix are merely the shifted versions of the first row -They are each shifted with respect to each other by "***One branch word***". If the information sequence ***u*** has a finite length, say ***L***, then ***G*** has ***L*** rows and ***n × (m +L)*** columns (or ***(m +L)*** branch word columns) and ***v*** has a length of ***n × (m +L)*** or a length of ***(m +L)*** branch words. Each branch word is of length '***n***'. Thus the Generator matrix ***G***, for the encoders of type shown in Fig 8.3 is written as:

$$G = \begin{bmatrix} g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & g_4^{(1)} g_4^{(2)} & \cdots & \cdots & \\ & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & g_4^{(1)} g_4^{(2)} & \cdots & \cdots \\ & & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & g_4^{(1)} g_4^{(2)} & \cdots \\ & & & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & g_4^{(1)} g_4^{(2)} \end{bmatrix} \quad \ldots\ldots\ldots (8.3)$$

(Blank places are zeros.)

The encoding equations in Matrix form is:

$$v = u \,.G \qquad (8.4)$$

Example 8.2:

For the information sequence of Example 8.1, the G matrix has 5 rows and 2(3 +5) =16 columns and we have

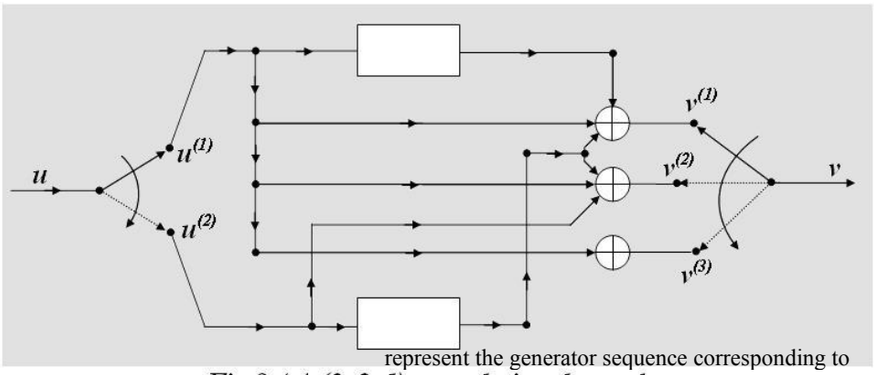
$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 &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

Performing multiplication, ***v = u G*** as per Eq (8.4), we get: ***v = (11, 01, 00, 01, 01, 00, 11)*** same as before.

As a second example of a convolutional encoder, consider the (***3, 2, 1***) encoder shown in

Fig.8.4. Here, as $k = 2$, the encoder consists of two $m = 1$ stage shift registers together with $n = 3$ modulo -2 adders and two multiplexers. The information sequence enters the encoder $k = 2$ bits at a time and can be written as $u = \{u_1^{(1)} u_1^{(2)}, u_2^{(1)} u_2^{(2)}, u_3^{(1)} u_3^{(2)} \ldots\}$ or as two separate input sequences:

$$u^{(1)} = \{u_1^{(1)}, u_2^{(1)}, u_3^{(1)} \ldots\} \text{ and } u^{(2)} = \{u_1^{(2)}, u_2^{(2)}, u_3^{(2)} \ldots\}.$$



represent the generator sequence corresponding to

Fig 8.4 A (3, 2, 1) convolutional encoder

There are three generator sequences corresponding to each input sequence. Letting $g_i^{(j)} = \{g_{i,1}^{(j)}, g_{i,2}^{(j)}, g_{i,3}^{(j)} \ldots g_{i,m+1}^{(j)}\}$ input i and output j. The generator sequences for the encoder are:

$$g_1^{(1)} = (1, 1), g_1^{(2)} = (1, 0), g_1^{(3)} = (1, 0)$$

$$g_2^{(1)} = (0, 1), g_2^{(2)} = (1, 1), g_2^{(3)} = (0, 0)$$

The encoding equations can be written as:

$$v^{(1)} = u^{(1)} * g_1^{(1)} + u^{(2)} * g_2^{(1)} \quad \ldots\ldots (8.5a)$$

$$v^{(2)} = u^{(1)} * g_1^{(2)} + u^{(2)} * g_2^{(2)} \quad \ldots\ldots (8.5b)$$

$$v^{(3)} = u^{(1)} * g_1^{(3)} + u^{(2)} * g_2^{(3)} \quad \ldots\ldots (8.5c)$$

The convolution operation implies that:

$$v_l^{(1)} = u_l^{(1)} + u_{l-1}^{(1)} + u_{l-1}^{(2)}$$

$$v_l^{(2)} = u_l^{(1)} + u_l^{(2)} + u_{l-1}^{(2)}$$

$$v_l^{(3)} = u_l^{(1)}$$

as can be seen from the encoding circuit.
After multiplexing, the code word is given by:

$$v = \{v_1^{(1)} v_1^{(2)} v_1^{(3)}, v_2^{(1)} v_2^{(2)} v_2^{(3)}, v_3^{(1)} v_3^{(2)} v_3^{(3)} \ldots\}$$

Example 8.3:

Suppose $\boldsymbol{u}$ = (*1 1 0 1 1 0*). Hence $\boldsymbol{u}^{(1)}$ = (*1 0 1*) and $\boldsymbol{u}^{(2)}$ = (*1 1 0*). Then

$$\boldsymbol{v}^{(1)} = (1\ 0\ 1) * (1,1) + (1\ 1\ 0) *(0,1) = (1\ 0\ 0\ 1)$$

$$\boldsymbol{v}^{(2)} = (1\ 0\ 1) * (1,0) + (1\ 1\ 0) *(1,1) = (0\ 0\ 0\ 0)$$

$$\boldsymbol{v}^{(3)} = (1\ 0\ 1) * (1,0) + (1\ 1\ 0) *(0,0) = (1\ 0\ 1\ 0)$$

$$\therefore \boldsymbol{v} = (1\ 0\ 1, 0\ 0\ 0, 0\ 0\ 1, 1\ 0\ 0).$$

The generator matrix for a (*3, 2, m*) code can be written as:

$$\boldsymbol{G} = \begin{bmatrix} g_{11}^{(1)} g_{11}^{(2)} & g_{11}^{(3)} & g_{12}^{(1)} & g_{12}^{(2)} & g_{13}^{(3)} & \cdots & g_{1,m+1}^{(1)} & g_{1,m+1}^{(2)} & g_{1,m+1}^{(3)} \\ g_{21}^{(1)} g_{21}^{(2)} & g_{21}^{(3)} & g_{22}^{(1)} & g_{22}^{(2)} & g_{22}^{(3)} & \cdots & g_{2,m+1}^{(1)} & g_{2,m+1}^{(2)} & g_{2,m+1}^{(3)} \\ & g_{11}^{(1)} & g_{11}^{(2)} & g_{11}^{(3)} & g_{12}^{(1)} & g_{12}^{(2)} & g_{12}^{(3)} & \cdots & \\ & g_{21}^{(1)} g_{21}^{(2)} g_{21}^{(3)} & & g_{22}^{(1)} g_{22}^{(2)} g_{22}^{(3)} & & & \cdots & & \\ & & & & & & & & \end{bmatrix} \quad \ldots\ldots (8.6)$$

The encoding equations in matrix form are again given by $\boldsymbol{v} = \boldsymbol{u}\ \boldsymbol{G}$. observe that each set of $\boldsymbol{k} = \boldsymbol{2}$ rows of $\boldsymbol{G}$ is identical to the preceding set of rows but shifted by $\boldsymbol{n} = \boldsymbol{3}$ places or one branch word to the right.

Example 8.4:

For the Example 8.3, we have

$$\boldsymbol{u} = \{u_1^{(1)} u_1^{(2)}, u_2^{(1)} u_2^{(2)}, u_3^{(1)} u_3^{(2)}\} = (1\ 1, 0\ 1, 1\ 0)$$

The generator matrix is:

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

Remember that the blank places in the matrix are all zeros.

Performing the matrix multiplication, $\boldsymbol{v} = \boldsymbol{u}\ \boldsymbol{G}$, we get: $\boldsymbol{v}$ = ***(101,000,001,100),*** again agreeing with our previous computation using discrete convolution.

This second example clearly demonstrates the complexities involved, when the number of input sequences are increased beyond k > 1, in describing the code. In this case, although the encoder contains k shift registers all of them need not have the same length. If $\boldsymbol{k_i}$ is the length of the ***i***-th shift register, then we define the encoder "***memory order, m***" by

$$m \triangleq \underset{1 \le i \le k}{Max}\, k_i \qquad \ldots\ldots\ldots\ldots\ldots \qquad (8.7)$$

(i.e. the maximum length of all ***k***-shift registers)

An example of a (***4, 3, 2***) convolutional encoder in which the shift register lengths are ***0***, ***1*** and ***2*** is shown in Fig 8.5.

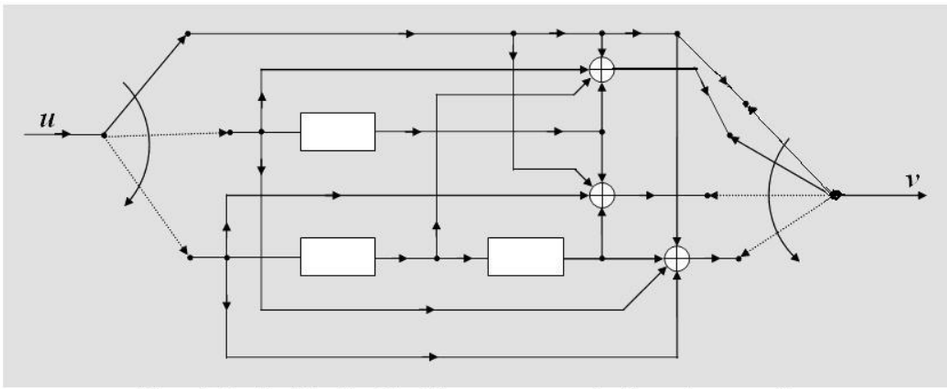


Fig 8.5 A (4, 3, 2) binary convolutional encoder

Since each information bit remains in the encoder up to (***m*** + ***1***) time units and during each time unit it can affect any of the ***n***-encoder outputs (which depends on the shift register connections) it follows that "***the maximum number of encoder outputs that can be affected by a single information bit***" is

$$n_A \triangleq n(m+1) \qquad \ldots\ldots\ldots\ldots\ldots \qquad (8.8)$$

' n_A' is called the 'constraint length" of the code. For example, the constraint lengths of the encoders of Figures 8.3, 8.4 and 8.5 are 8, 6 and 12 respectively. Some authors have defined the constraint length (For example: Simon Haykin) as the number of shifts over which a single message bit can influence the encoder output. In an encoder with an ***m***-stage shift register, the " ***memory***" of the encoder equals ***m***-message bits, and the constraint length $n_A = (m + 1)$. However, we shall adopt the definition given in Eq (8.8).

The number of shifts over which a single message bit can influence the encoder output is usually denoted as ***K.*** For the encoders of Fig 8.3, 8.4 and 8.5 have values of ***K*** = ***4, 2*** and ***3*** respectively. The encoder in Fig 8.3 will be accordingly labeled as a ' ***rate 1/2, K = 4'*** convolutional encoder. The term ***K*** also signifies the number of branch words in the encoder's impulse response.

Turning back, in the general case of an (***n, k, m***) code, the generator matrix can be put in the form:

$$G = \begin{bmatrix} G_1 & G_2 & G_3 & \cdots & G_m & G_{m+1} & & \\ & G_1 & G_2 & \cdots & G_{m-1} & G_m & G_{m+1} & \\ & & G_1 & \cdots & G_{m-2} & G_{m-1} & G_m & G_{m+1} \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \qquad \ldots\ldots\ldots\ldots \qquad (8.9)$$

Where each G_i is a ($k \times n$) sub matrix with entries as below:

Notice that each set of k-rows of G are identical to the previous set of rows but shifted n-places to the right. For an information sequence $u = (u_1, u_2 \ldots)$ where $u_i = \{u_i^{(1)}, u_i^{(2)} \ldots u_i^{(k)}\}$, the code word is $v = (v_1, v_2 \ldots)$ where $v_i = (v_i^{(1)}, v_i^{(2)} \ldots v_i^{(n)})$ and $v = u\,G$. Since the code word is a linear combination of rows of the G matrix it follows that an (n, k, m) convolutional code is a linear code.

Since the convolutional encoder generates n-encoded bits for each k-message bits, we define $R = k/n$ as the "***code rate***". However, an information sequence of finite length L is encoded into a code word of length $n \times (L + m)$, where the final $n \times m$ outputs are generated after the last non zero information block has entered the encoder. That is, an information sequence is terminated with all zero blocks in order to clear the encoder memory. (To appreciate this fact, examine 'the calculations of $v_l^{(j)}$ for the Example 8.l and 8.3). The terminating sequence of m-zeros is called the "***Tail of the message***". Viewing the convolutional-code as a linear block code, with generator matrix G, then the block code rate is given by $kL/n(L + m)$ - the ratio of the number of message bits to the length of the code word. If $L >> m$, then, $L/(L + m) \approx 1$ and the block code rate of a convolutional code and its rate when viewed as a block code would appear to be same. Infact, this is the normal mode of operation for convolutional codes and accordingly we shall not distinguish between the rate of a convolutional code and its rate when viewed as a block code. On the contrary, if ' L' were small, the effective rate of transmission indeed is $kL/n\,(L + m)$ and will be below the block code rate by a fractional amount:

$$\frac{k/n - kL/n(L+m)}{k/n} = \frac{m}{L+m} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (8.11)$$

and is called "***fractional rate loss***". Therefore, in order to keep the fractional rate loss at a minimum (near zero), ' L' is always assumed to be much larger than ' m'. For the information 'sequence of Example 8.1, we have $L = 5$, $m = 3$ and ***fractional rate loss*** $= 3/8 = 37.5\%$. If L is made 1000, the fractional rate loss is only $3/1003 \approx 0.3\%$.

8.3 Encoding of Convolutional Codes; Transform Domain Approach:

In any linear system, we know that the time domain operation involving the convolution integral can be replaced by the more convenient transform domain operation, involving polynomial multiplication. Since a convolutional encoder can be viewed as a 'linear time invariant finite state machine, we may simplify computation of the adder outputs by applying appropriate transformation. As is done in cyclic codes, each 'sequence in the encoding equations can' be replaced by a corresponding polynomial and the convolution operation replaced by polynomial multiplication. For example, for a $(2, 1, m)$ code, the encoding equations become:

$$v^{(1)}(X) = u(X)\, g^{(1)}(X) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (8.12a)$$

$$v^{(2)}(X) = u(X)\, g^{(2)}(X) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (8.12b)$$

Where $u(X) = u_1 + u_2 X + u_3 X^2 + \ldots$ is the information polynomial,

$v^{(1)}(X) = v_1^{(1)} + v_2^{(1)} X + v_3^{(1)} X^2 + \ldots\ldots$, and

$$v^{(2)}(X) = v_1^{(2)} + v_2^{(2)}X + v_3^{(2)}X^2 + ...$$

are the encoded polynomials.

$$g^{(1)}(X) = g_1^{(1)} + g_2^{(1)}X + g_3^{(1)}X^2 + ..., \text{ and}$$

$$g^{(2)}(X) = g_1^{(2)} + g_2^{(2)}X + g_3^{(2)}X^2 + ...$$

are the "generator polynomials" of' the code; and all operations are modulo-2. After multiplexing, the code word becomes:

$$v(X) = v^{(1)}(X^2) + X\,v^{(2)}(X^2) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (8.13)$$

The indeterminate '***X***' can be regarded as a "***unit-delay operator***", the power of ***X*** defining the number of time units by which the associated bit is delayed with respect to the initial bit in the sequence.

Example 8.5:

For the (***2, 1, 3***) encoder of Fig 8.3, the impulse responses were: $g^{(1)} = (1,0,1,1)$, and $g^{(2)} = (1,1,1,1)$

The generator polynomials are: $g^{(l)}(X) = 1 + X^2 + X^3$, and $g^{(2)}(X) = 1 + X + X^2 + X^3$

For the information sequence ***u = (1, 0, 1, 1, 1);*** the information polynomial is: $u(X) = 1+X^2+X^3+X^4$

The two code polynomials are then:

$$v^{(1)}(X) = u(X)\,g^{(l)}(X) = (1 + X^2 + X^3 + X^4)(1 + X^2 + X^3) = 1 + X^7$$

$$v^{(2)}(X) = u(X)\,g^{(2)}(X) = (1 + X^2 + X^3 + X^4)(1 + X + X^2 + X^3) = 1 + X + X^3 + X^4 + X^5 + X^7$$

From the polynomials so obtained we can immediately write:

$v^{(1)} = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1)$, and $v^{(2)} = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1)$

Pairing the components we then get the code word ***v = (11, 01, 00, 01, 01, 01, 00, 11).***

We may use the multiplexing technique of Eq (8.13) and write:

$v^{(1)}(X^2) = 1 + X^{14}$ and $v^{(2)}(X^2) = 1+X^2+X^6+X^8+X^{10}+X^{14}$; $Xv^{(2)}(X^2) = X + X^3 + X^7 + X^9 + X^{11} + X^{15}$;

and the code polynomial is: $v(X) = v^{(1)}(X^2) + X\,v^{(2)}(X^2) = 1 + X + X^3 + X^7 + X^9 + X^{11} + X^{14} + X^{15}$

Hence the code word is: ***v = (1 1, 0 1, 0 0, 0 1, 0 1, 0 1, 0 0, 1 1);*** this is exactly the same as obtained earlier.

The generator polynomials of an encoder can be determined directly from its circuit diagram. Specifically, the co-efficient of X^l is a '***1***' if there is a "***connection***" from the ***l***-th shift register stage to the input of the adder of interest and a '***0***' otherwise. Since the last stage of the shift register in an

(***n, l***) code must be connected to at least one output, it follows that at least one generator polynomial should have a degree equal to the shift register length '***m***', i.e.

$$m = \underset{1 \le j \le n}{Max} \left\{ deg\ g^{(j)}(X) \right\} \quad \ldots\ldots\ldots\ldots\ldots \quad (8.14)$$

In an (***n, k***) code, where ***k > 1***, there are ***n***-generator polynomials for each of the ***k***-inputs, each set representing the connections from one of the shift registers to the ***n***-outputs. Hence, the length K_l of the ***l***-th shift register is given by:

$$K_l = \underset{1 \le j \le n}{Max} \left\{ deg\ g_l^{(j)}(X) \right\}, 1 \le l \le k \quad \ldots\ldots\ldots\ldots\ldots \quad (8.15)$$

Where $g_l^{(j)}(X)$ is the generator polynomial relating the ***l***-th input to the ***j***-th output and the encoder memory order ***m*** is:

$$m = \underset{1 \le l \le k}{Max} K_l = \underset{1 \le l \le k}{Max} \left\{ \underset{1 \le j \le n}{Max} \left\{ deg\ g_l^{(j)}(X) \right\} \right\} \quad \ldots\ldots\ldots\ldots \quad (8.16)$$

Since the encoder is a linear system and $u^{(l)}(X)$ represents the ***l***-th input sequence and $v^{(j)}(X)$ represents the ***j***-th output sequence the generator polynomial $g_l^{(j)}(X)$ can be regarded as the 'encoder transfer function' relating the input - ***l*** to the output – ***j.*** For the ***k***-input, ***n***- output linear system there are a total of ***k×n*** transfer functions which can be represented as a (***k*** × ***n***) "***transfer function matrix***".

$$G(X) = \begin{matrix} g_1^{(1)}(X), & g_1^{(2)}(X), & \ldots & g_1^{(n)}(X) \\ g_2^{(1)}(X), & g_2^{(2)}(X), & \ldots & g_2^{(n)}(X) \\ \vdots & \vdots & & \vdots \\ g_k^{(1)}(X), & g_k^{(2)}(X), & \ldots & g_k^{(n)}(X) \end{matrix} \quad \ldots\ldots\ldots\ldots \quad (8.17)$$

Using the transfer function matrix, the encoding equations for an (***n, k, m***) code can be expressed as

$$V(X) = U(X)\ G(X) \quad \ldots\ldots\ldots\ldots\ldots \quad (8.18)$$

$U(X) = [u^{(1)}(X), u^{(2)}(X) \ldots u^{(k)}(X)]$ is the ***k***-vector, representing the information polynomials, and. $V(X) = [v^{(1)}(X), v^{(2)}(X) \ldots v^{(n)}(X)]$ is the ***n***-vector representing the encoded sequences. After multiplexing, the code word becomes:

$$v(X) = v^{(1)}(X^n) + X\, v^{(2)}(X^n) + X^2 v^{(3)}(X^n) + \ldots + X^{n-l} v^{(n)}(X^n) \quad \ldots\ldots\ldots\ldots \quad (8.19)$$

Example 8.6:

For the encoder of Fig 8.4, we have:

$g_1^{(1)}(X) = 1 + X, \quad g_2^{(1)}(X) = X$

$g_{1(2)}(X) = 1, \quad g_{2(2)}(X) = 1 + X$

$g_{1(3)}(X) = 1, \quad g_{2(3)}(X) = 0$

$$\therefore G(X) = \begin{matrix} 1+X & 1 & 1 \\ X & 1+X & 0 \end{matrix}$$

For the information sequence $u^{(1)} = (1\ 0\ 1)$, $u^{(2)} = (1\ 1\ 0)$, the information polynomials are: $u^{(1)}(X) = 1 + X^2$, $u^{(2)}(X) = 1 + X$

Then $V(X) = [v^{(1)}(X), v^{(2)}(X), v^{(3)}(X)]$

$$= [1 + X^2,\ 1 + X] \begin{matrix} 1+X & 1 & 1 \\ X & 1+X & 0 \end{matrix} = [1+X^3,\ 0,\ 1+X^2]$$

Hence the code word is:

$$v(X) = v^{(1)}(X^3) + Xv^{(2)}(X^3) + X^2v^{(3)}(X^3)$$

$$= (1 + X^9) + X(0) + X^2(1 + X^6)$$

$$= 1 + X^2 + X^8 + X^9$$

$$\therefore v = (1\ 0\ 1,\ 0\ 0\ 0,\ 0\ 0\ 1,\ 1\ 0\ 0).$$

This is exactly the same as that obtained in Example 8.3.From Eq (8.17) and (8.18) it follows that:

8.5.1 State Diagrams:

The state of an encoder is defined as its shift register contents. For an (n, k, m) code with $k > 1$, i-th shift register contains 'K_i' previous information bits. Defining $K = \sum_{i=1}^{k} K_i$ as the total encoder - memory (m - represents the memory order which we have defined as the maximum length of any shift register), the encoder state at time unit T', when the encoder inputs are, $\{u_l^{(1)}, u_l^{(2)} \ldots u_l^{(k)}\}$, are the binary k-tuple of inputs:

$\{u_{l-1}^{(1)} u_{l-2}^{(1)}, u_{l-3}^{(1)} \ldots u_{l-k}^{(1)}; u_{l-1}^{(2)}, u_{l-2}^{(2)}, u_{l-3}^{(2)} \ldots u_{l-k}^{(2)}; \ldots; u_{l-1}^{(k)} u_{l-2}^{(k)}, u_{l-3}^{(k)} \ldots u_{l-k}^{(k)}\}$, and there are a total of 2^k different possible states. For a $(n, 1, m)$ code, $K = K_1 = m$ and the encoder state at time unit l is simply $\{u_{l-1}, u_{l-2} \ldots u_{l-m}\}$.

Each new block of k-inputs causes a transition to a new state. Hence there are 2^k branches leaving each state, one each corresponding to the input block. For an $(n, 1, m)$ code there are only two branches leaving each state. On the state diagram, each branch is labeled with the k-inputs causing the transition and the n-corresponding outputs. The state diagram for the convolutional encoder of Fig 8.3 is shown in Fig 8.10. A ***state table*** would be, often, more helpful while drawing the state diagram and is as shown.

State table for the (2, 1, 3) encoder of Fig 8.3

State	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Binary Description	*000*	*100*	*010*	*110*	*001*	*101*	*011*	*111*

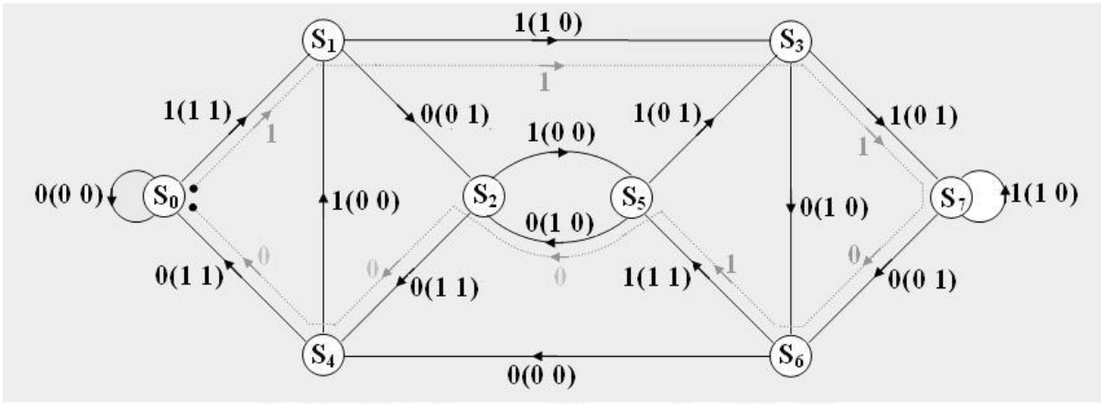


Fig 8.10 State diagram of the encoder of Fig 8.3

Recall (or observe from Fig 8.3) that the two out sequences are:

$$v^{(1)} = u_l + u_{l-2} + u_{l-3} \quad \text{and}$$
$$v^{(2)} = u_l + u_{l-1} + u_{l-2} + u_{l-3}$$

Till the reader, gains some experience, it is advisable to first prepare a transition table using the output equations and then translate the data on to the state diagram. Such a table is as shown below:

State transition table for the encoder of Fig 8.3

Previous State	*Binary Description*	*Input*	*Next State*	*Binary Description*	u_l	u_{l-1}	u_{l-2}	u_{l-3}	*Output*
S_0	0 0 0	0	S_0	0 0 0	0	0	0	0	0 0
		1	S_1	1 0 0	1	0	0	0	1 1
S_1	1 0 0	0	S_2	0 1 0	0	1	0	0	0 1
		1	S_3	1 1 0	1	1	0	0	1 0
S_2	0 1 0	0	S_4	0 0 1	0	0	1	0	1 1
		1	S_5	1 0 1	1	0	1	0	0 0
S_3	1 1 0	0	S_6	0 1 1	0	1	1	0	1 0
		1	S_7	1 1 1	1	1	1	0	0 1
S_4	0 0 1	0	S_0	0 0 0	0	0	0	1	1 1
		1	S_1	1 0 0	1	0	0	1	0 0
S_5	1 0 1	0	S_2	0 1 0	0	1	0	1	1 0
		1	S_3	1 1 0	1	1	0	1	0 1
S_6	0 1 1	0	S_4	0 0 1	0	0	1	1	0 0
		1	S_5	1 0 1	1	0	1	1	1 1
S_7	1 1 1	0	S_6	0 1 1	0	1	1	1	0 1
		1	S_7	1 1 1	1	1	1	1	1 0

For example, if the shift registers were in state S_5, whose binary description is *101*, an input ' *1*' causes this state to change over to the new state S_3 whose binary description is *110* while producing

an output (***0 1***). Observe that the inputs causing the transition are shown first, followed by the corresponding output sequences shown with in parenthesis.

Assuming that the shift registers are initially in the state S_0 (the all zero state) the code word corresponding to any information sequence can be obtained by following the path through the state diagram determined by the information sequence and noting the corresponding outputs on the branch labels. Following the last nonzero block, the encoder is returned to state S_0 by a sequence of m-all-zero block appended to the information sequence. For example, in Fig 8.10, if $u = (11101)$, the code word is $v = (11, 10, 01, 01, 11, 10, 11, 10)$ the path followed is shown in thin gray lines with arrows and the input bit written along in thin gray. The $m = 3$ zeros appended are indicated in gray which is much lighter compared to the information bits.

Example 8.12: A (*2, 1, 2*) Convolutional Encoder*:*

Consider the encoder shown in Fig 8.15. ***We shall use this example for discussing further graphical representations viz. Trees, and Trellis.***

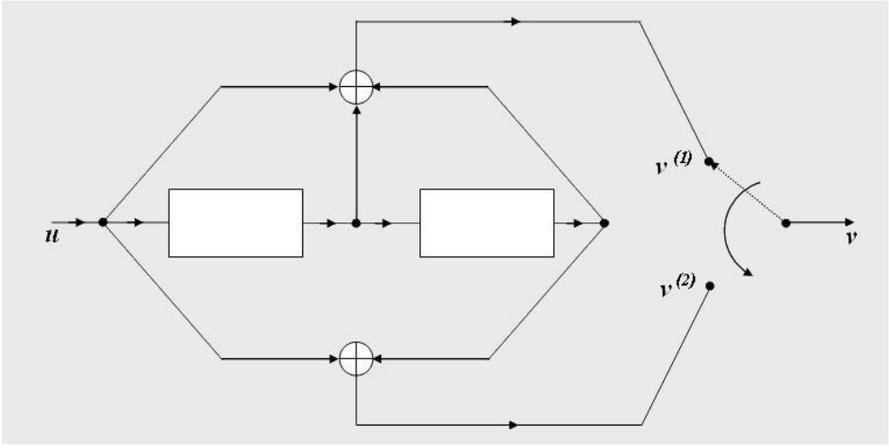


Fig 8.15 A (2, 1, 2) Convolutional encoder

For this encoder we have: $v_l^{(1)} = u_l + u_{l-1} + u_{l-2}$ and $v_l^{(2)} = u_l + u_{l-2}$
The state transition table is as follows.

State transition table for the (2, 1, 2) convolutional encoder of Example 8.12

Previous state	*Binary description*	*Input*	*Next State*	*Binary description*	u_l	u_{l-1}	u_{l-2}	*Output*
S0	*0 0*	*0*	*S0*	*0 0*	*0*	*0*	*0*	*0 0*
		1	*S1*	*1 0*	*1*	*0*	*0*	*1 1*
S1	*1 0*	*0*	*S2*	*0 1*	*0*	*1*	*0*	*1 0*
		1	*S3*	*1 1*	*1*	*1*	*0*	*0 1*
S2	*0 1*	*0*	*S0*	*0 0*	*0*	*0*	*1*	*1 1*
		1	*S1*	*1 0*	*1*	*0*	*1*	*0 0*
S3	*1 1*	*0*	*S2*	*0 1*	*0*	*1*	*1*	*0 1*
		1	*S3*	*1 1*	*1*	*1*	*1*	*1 0*

The state diagram and the augmented state diagram for computing the 'complete path enumerator function' for the encoder are shown in Fig 8.16.

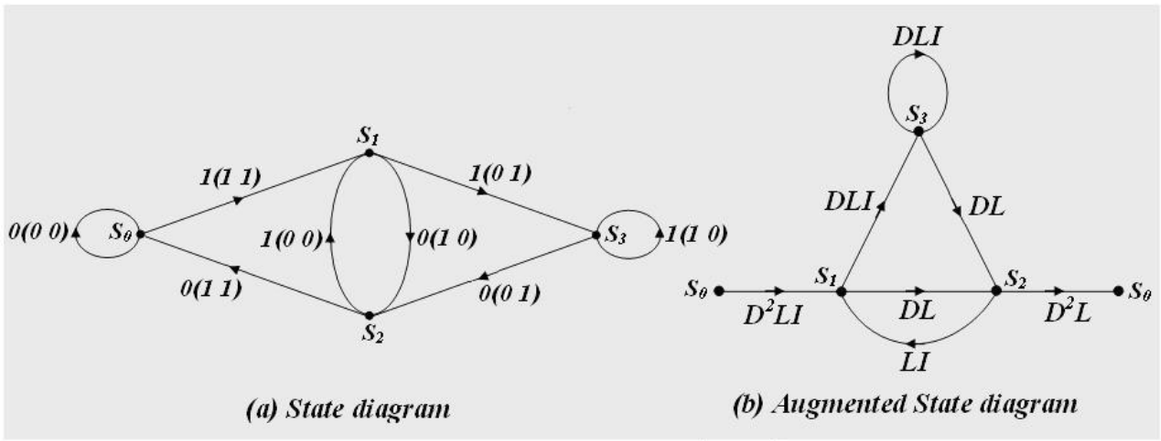


Fig 8.16 State diagram for the (2, 1, 2) encoder

8.5.3 Catastrophic Code:

We shall re-consider the catastrophic codes considered earlier. The state diagram of a (***2, 1, 2***) code is shown in Fig 8.17. Notice that for a catastrophic code "***the state diagram consists of a loop of zero weight other than the self loop around the state S_0***".

It is the characteristic of a catastrophic code that an information sequence of infinite weight produces a finite weight code word.

In a non-catastrophic code which contains no zero weight loops other than the self loop around the all zero state S_0, all infinite weight information sequences must generate infinite weight code words, and minimum weight code word always has a finite length.

The best achievable d_{free} for a convolutional code with a given rate and constraint length has not been determined exactly. However, results are available with respect to the lower and upper bounds on d_{free} for the best code, obtained using random coding approach. It is observed that more free distance is available with non-systematic codes of a given rate and constraint length compared to the systematic codes and thus has important consequences when a code with large d_{free} must be selected for use with either the Viterbi or Sequential decoding.

8.5.4TreeandTrellisDiagrams:

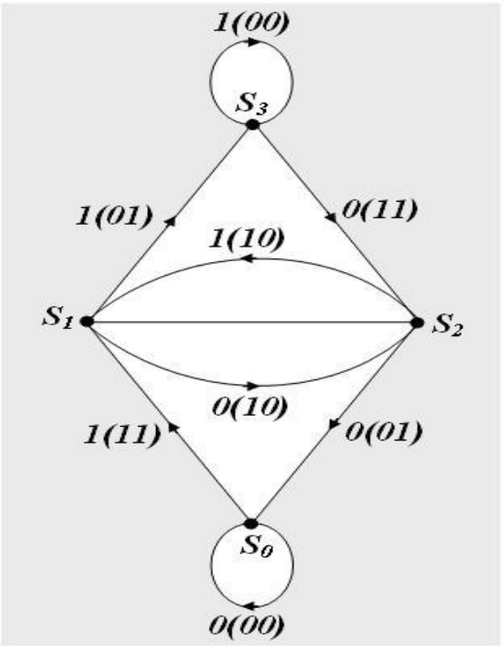


Fig 8.17 State diagram of a (2, 1, 2) catastrophic Code

Let us now consider other graphical means of portraying convolutional codes. The state diagram can be re-drawn as a 'Tree graph'. The convention followed is: If the input is a '***0***', then the upper path is followed and if the input is a '***1***', then the lower path is followed. A vertical line is called a '***Node***' and a horizontal line is called '***Branch***'. The output code words for each input bit are shown on the branches. The encoder output for any information sequence can be traced through the tree paths. The tree graph for the (***2, 1, 2***) encoder of Fig 8.15 is shown in Fig 8.18. The state transition table can be conveniently used in constructing the tree graph.

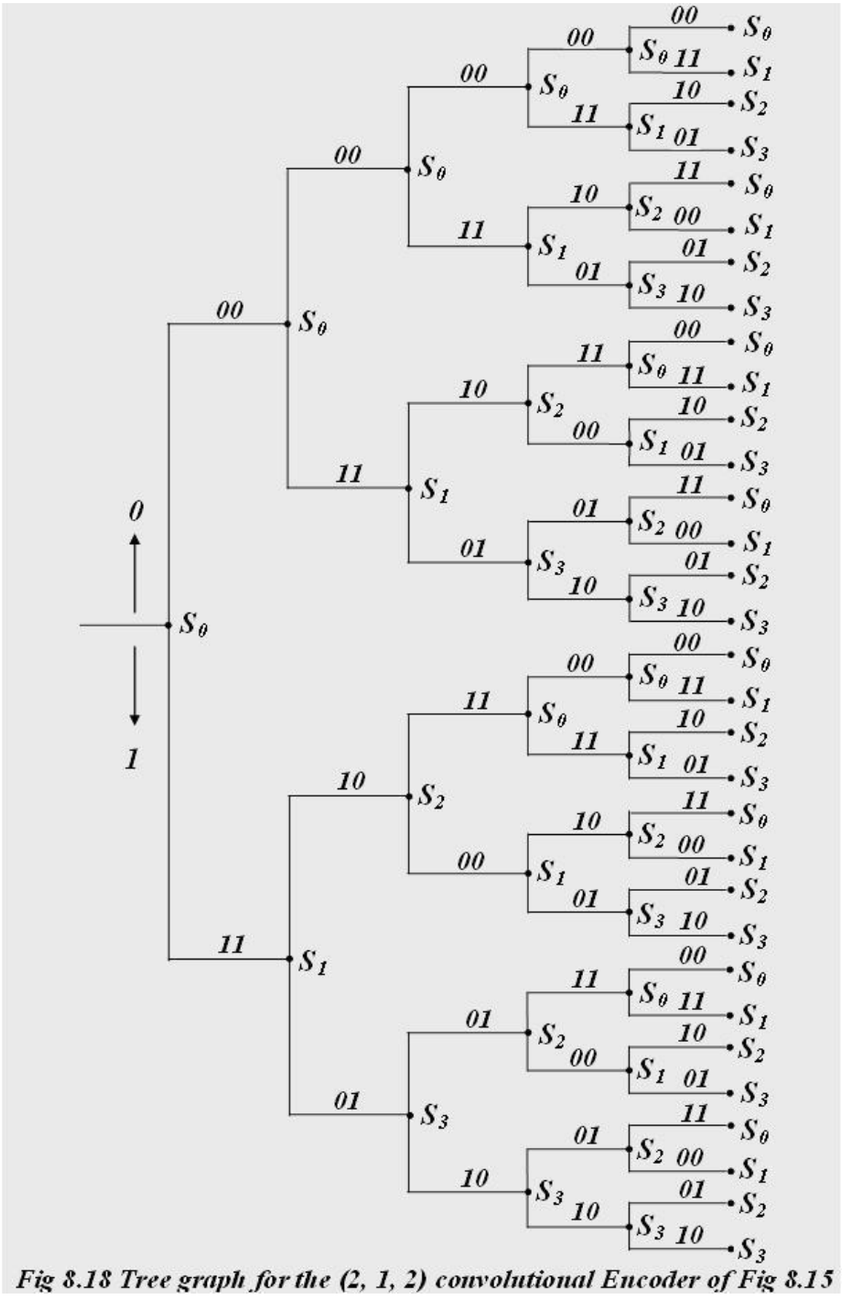


Fig 8.18 Tree graph for the (2, 1, 2) convolutional Encoder of Fig 8.15

Following the procedure just described we find that the encoded sequence for an information sequence (***10011***) is (***11, 10, 11, 11, 01***) which agrees with the first ***5*** pairs of bits of the actual

encoded sequence. Since the encoder has a memory = ***2*** we require two more bits to clear and re-set the encoder. Hence to obtain the complete code sequence corresponding to an information sequence of length ***kL***, the tree graph is to extended by ***n(m-l)*** time units and this extended part is called the "***Tail of the tree***", and the ***2kL*** right most nodes are called the "***Terminal nodes***" of the tree. Thus the extended tree diagram for the (***2, 1, 2***) encoder, for the information sequence (***10011***) is as in Fig 8.19 and the complete encoded sequence is (***11, 10, 11, 11, 01, 01, 11***).

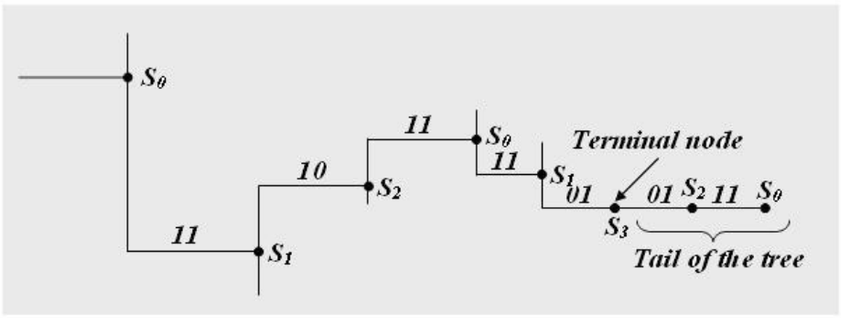


Fig 8.19 Illustration of the 'Tail of the tree'

At this juncture, a very important clue for the student in drawing tree diagrams neatly and correctly, without wasting time appears pertinent. As the length of the input sequence ***L*** increases the number of right most nodes increase as 2^L. Hence for a specified sequence length, ***L***, compute 2^L. Mark 2^L equally spaced points at the rightmost portion of your page, leaving space to complete the ***m*** tail branches. Join two points at a time to obtain 2^{L-l} nodes. Repeat the procedure until you get only one node at the left most portion of your page. The procedure is illustrated diagrammatically in Fig 8.20 for ***L = 3.*** Once you get the tree structure, now you can fill in the needed information either looking back to the state transition table or working out logically.

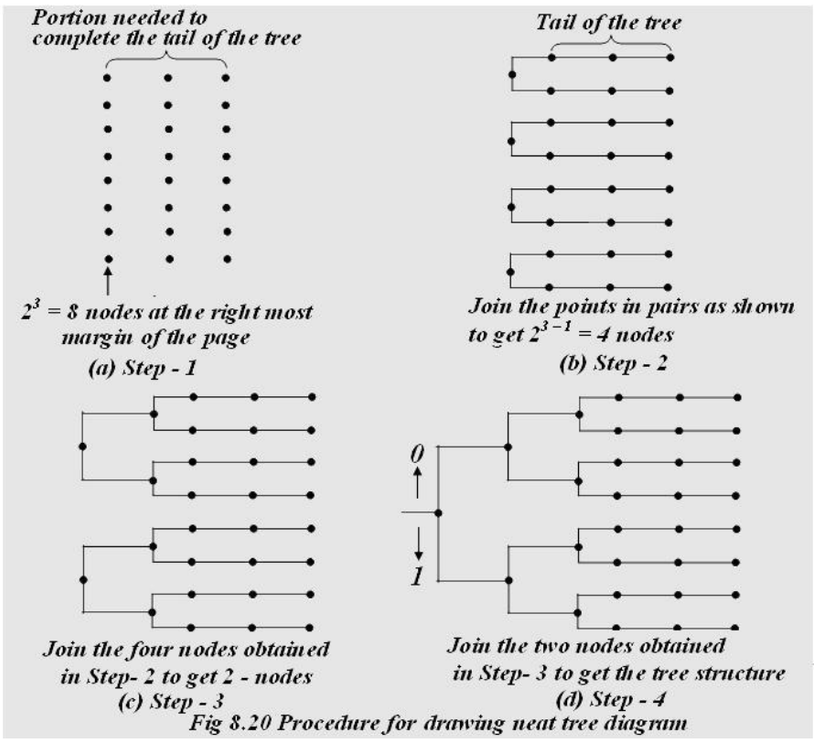


Fig 8.20 Procedure for drawing neat tree diagram

From Fig 8.18, observe that the tree becomes "***repetitive***' after the first three branches. Beyond the third branch, the nodes labeled S_0 are identical and so are all the other pairs of nodes that are identically labeled. Since the encoder has a memory ***m = 2,*** it follows that when the third information bit enters the encoder, the first message bit is shifted out of the register. Consequently, after the third branch the information sequences (***000u_3u_4---***) and (***100u_3u_4---***) generate the same code symbols and the pair of nodes labeled S_0 may be joined together. The same logic holds for the other nodes.

Accordingly, we may collapse the tree graph of Fig 8.18 into a new form of Fig 8.21 called a "***Trellis***". It is so called because Trellis is a tree like structure with re-merging branches (You will have seen the trusses and trellis used in building construction).

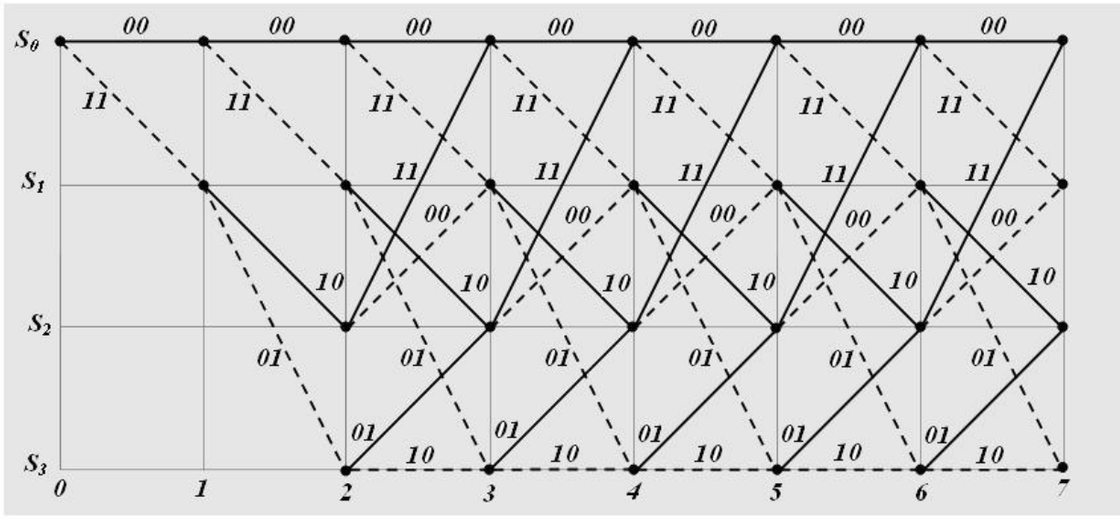


Fig 8.21 Trellis diagram for the encoder of Fig 8.15

The Trellis diagram contain (***L + m + 1***) time units or levels (or depth) and these are labeled from ***0*** to (***L + m***) (***0 to 7*** for the case with ***L = 5*** for encoder of Fig 8.15 as shown in Fig8.21. The convention followed in drawing the Trellis is that "***a code branch produced by an input '0' is drawn as a solid line while that produced by an input '1' is shown by dashed lines***". The code words produced by the transitions are also indicated on the diagram. Each input sequence corresponds to a specific path through the trellis. The reader can readily verify that the encoder output corresponding to the sequence ***u = (10011)*** is indeed ***v = (11, 10, 11, 11, 01, 01, 11)*** the path followed being as shown in Fig 8.22.

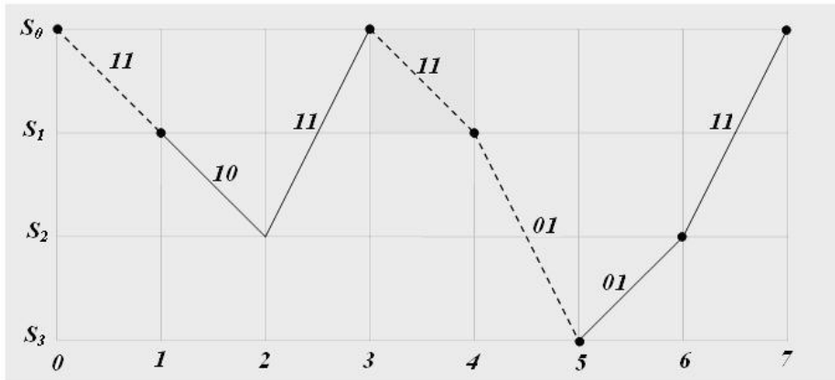


Fig 8.22 Path followed for the input sequence u = (10011)

A trellis is more instructive than a tree graph in that the finite state behaviour of the encoder is explicitly brought-out. Assuming that the encoder always starts in state S_0 and returns to S_0, the first m - time units (levels or depth) correspond to the encoder's departure from S_0 and the last m-levels corresponds to its return to S_0. Clearly, not all the states can be reached in these two portions of the Trellis. However, in the centre portion of the Trellis, all states are possible and each level (time unit) contains a replica of the state diagram and is shown in Fig 8.23. There are two branches leaving and entering each state.

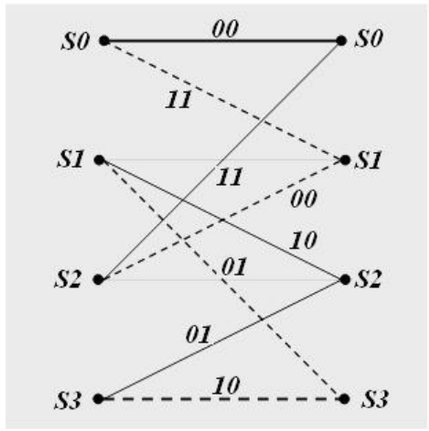


Fig 8.23 A portion of the central Part of the trellis of Fig 8.20

In the general case of an (n, k, m) code and an information sequence of length kL, there are 2^k branches leaving and entering each state and 2^{kL} distinct paths through the trellis corresponding to the 2^{kL} code words.

The following observations can be made from the Trellis diagram

1. There are no fundamental paths at distance *1*, *2* or *3* from the all zero path.

2. There is a single fundamental path at distance *5* from the all zero path. It diverges from the all-zero path three branches back and it differs from the all-zero path in the single input bit.

3. There are two fundamental paths at a distance *6* from the all zero path. One path diverges from the all zero path four branches back and the other five branches back. Both paths differ from the all zero path in two input bits. The above observations are depicted in Fig 8.24(*a*).

4. There are four fundamental paths at a distance *7* from the all-zero path. One path diverges from the all zero path five branches back, two other paths six branches back and the fourth path diverges seven branches back as shown in Fig 8.24(*b*). They all differ from the all zero path in three input bits. This information can be compared with those obtained from the complete path enumerator function found earlier.