**LABORATORY MANUAL**

**FOR**

**Information Theory and Coding Techniques**

**(TE E&TC SEM II)**



**Department of Electronics and Telecommunication Engineering**

**International Institute of Information Technology**

**Hinjewadi,Pune - 411 057**

[**www.isquareit.edu.in**](http://www.isquareit.edu.in)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Work Load** | | **Exam Schemes** | | | |
| **Practical** | | **Term Work** | **Practical** | | **Oral** |
| 02 hrs per week | | -- | 50 | | --- |
| **List of Assignments** | | | | | |
| **Sr. No.** | **Title of Assignment** | | | **Time Span**  **(No. of weeks)** | |
|  | Write a program for determination of various entropies and mutual information of a given channel. Test various types of channel such as  a) Noise free channel. b) Error free channel  c) Binary symmetric channel d) Noisy channel  Compare channel capacity of above channels. | | | **2** | |
|  | Write a program for generation and evaluation of variable length source coding using C/MATLAB (Any 2)  a) Shannon – Fano coding and decoding  b) Huffman Coding and decoding  c) Lempel Ziv Coding and decoding | | | **4** | |
|  | Write a Program for coding & decoding of Linear block codes. | | | **1** | |
|  | Write a Program for coding & decoding of Cyclic codes. | | | **1** | |
|  | Write a program for coding and decoding of Convolutional codes | | | **1** | |
|  | Write a program for coding and decoding of BCH and RS codes. | | | **1** | |
|  | Implementation of ARQ Technique | | | **1** | |
|  | Study of Networking Components & LAN | | | **1** | |

Text Book:

1. Ranjan Bose, “Information Theory coding and Cryptography”, McGraw-Hill Publication, 2nd Edition
2. J C Moreira, P G Farrell, “Essentials of Error-Control Coding”, Wiley Student Edition

References

1. Bernad Sklar, “Digital Communication Fundamentals & applications”, 2nd Ed. Pearson Education.
2. Shu lin and Daniel j, Cistello jr., “Error control Coding” Pearson, 2nd Edition.
3. Todd Moon, “Error Correction Coding : Mathematical Methods and Algorithms”, Wiley Publication
4. Khalid Sayood, “Introduction to data compression”, Morgan Kaufmann Publishers

**ASSIGNMENT NO. 1**

**TITLE: ENTROPY AND MUTUAL INFORMATION DETERMINATION FOR GIVEN CHANNEL**

**PROBLEM STATEMENT:** Write C/MATLAB program to compute entropy and mutual information for following channels.

1. Noise free channel
2. Error free channel
3. Binary symmetric channel
4. Noisy channel

**OBJECTIVE:**

1. To understand the concept Entropy and mutual information.
2. To understand algorithm for computation of Entropy and mutual information.

**THEORY:**

**INFORMATION:**

The probability denotes likelihood or the certainty of occurrence of any event. A less probable event is rarer and so it contains more information. Thus, if an event of lower probability occurs, it conveys more information than the occurrence of an event of larger probability. If ‘p’ is the probability of occurrence of the message symbol and ‘I’ is the information received from the message, then;

Information, 

If the base is 2, unit of information is bit.

**Table for conversion of information bit:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No. | Unit | Base 2 | Base e | Base10 |
| 1. | Bit | Bit |  |  |
| 2. | Nat |  | Nat | 1Nat=1/(log 1e10) |

**Entropy (Average Information):**

Suppose there are `m` different messages m1,m2 ,….,mm  having probabilities p1,p2….pm. suppose a sequence of L message is transmitted. If L is very large, then we can say that messages of m are transmitted.





. . .

. . .



Thus, the total information due to sequence of L message will be…



Average information or Entropy = total information no. of messages.

Entropy H(X) =IL (total) = 

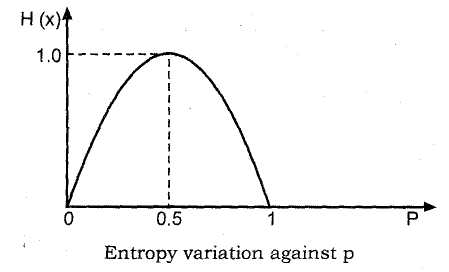


Fig. 1.1: Entropy Variation against p

It can be observed from graph that -

(i) H(X) is non-negative.

(ii) H(X) is zero only for p = 0 and p = 1 as there is no uncertainty.

(iii) H(X) is maximum at p = 1/2

**Properties of Information:**

The important properties of the information conveyed by a message are as follows-

1. The information contents of a message increases with the decrease in value of its probability (Pr). This means that the most unexpected event will contain maximum information.
2. Information is a continuous function of probability (PX).
3. Total information of two or more statically independent message signals events is equal to the sum of information contents of the individual messages, i.e.I total =I1+I2+… Where, I­total = total information
4. Information contained in a message can either be zero(0), if the probability of message is 1 or greater than 0, but it can never be a negative value , i.e. there is absolutely no loss of information.

I (Sk)>=0 if 0 < Pk < 1

1. If Sk and Si are two statistically independent events, then information contained in the combined event is equal to the sum of the information contained in the individual events I (Sk SI) = I (SK) + I (SI)
2. If we are absolutely certain of the outcome of an event even before it occurs that is probability of an event is one, then there is no information gained.

I (SK) = 0 if pk= 1

**Discrete Memoryless channel:**

Consider Discrete Memoryless channel

Let X : input to channel and Y : output of channel

Y being the noisy version of X

X and Y are random variables

x1, x2,…..xm : message belonging to input alphabet X

y1, y2,…..yn : message belonging to output alphabet Y

p(xj) : Probability of message xj where j = 1,2,3,…m

p(yk) : Probability of message yk where k = 1,2,3,…n

* Since X and Y are independent
* Transition Probabilities :
* P(yk/xj) :Probability of getting output as yk when input is xj

**Channel Matrix**



With the fundamental property 

Joint Probability Distribution of random variable X and Y is given by



Relation between Joint probability and Channel matrix



Probability distribution of output random variable Y is given by



This means that probability of yk is probability that yk and x1 occurs OR probability that yk and x2 occurs OR probability that yk and x3 occurs …. Up to yk and xm occurs



1. Probability of Transmitted symbols (Also called Marginal Probability of input variable or Priori Probability)



1. Probability of Received symbols (Also called Marginal Probability of output variable or Posteriori Probability)



1. Probability that symbol xj transmitted and yk is received (Also called joint Probability)



1. Probability that symbol yk is received and given xj transmitted (Also called conditional Probability OR Transition Probability)



1. Probability that symbol xj transmitted, given yk is received



**Marginal Entropy of Source / input X :**

**Marginal Entropy of Sink / output Y :**

**Joint Entropy of input X and output Y :**

|  |  |
| --- | --- |
| Entropy | Interpretation |
| H(X) | Average information per message at transmitter or entropy of transmitter |
| H(Y) | Average information per message at Receiver or entropy of Receiver |
| H(X|Y) | One of xj is transmitted with given probability and specific yk is received. |
| H(Y|X) | It is measure of information about the receiver where it is known that X is transmitted |
| H(X,Y) | Average Information per pair of transmitted and received messages or average uncertainty of communication system as a whole |

Information content provided by the occurrence of the event Y =*y* about event X = *x* is defined as

Where is called Mutual information between x and y

Mutual information between Random Variable X and Y is average of 



**EXAMPLE:**

Find out all entropies: H (X), H (Y), H (X, Y), H (X/Y) and H (Y/X).

The probability matrix is



**Solution:**

Summation of all elements of matrix is a joint probability matrix P (X/Y).

Summation of each row gives P (X).

P (X0) = 0.4 P (X1) = 0.3 P (X2) = 0.3

Summation of each column gives P (Y).

P (Y0) = 0.3 P (Y1) = 0.3 P (Y2) = 0.4

****

****



H (X/Y) = H (X, Y) – H (Y) = 3.12 – 1.57 = 1.55 bits/message

H (Y/X) = H (X, Y) – H (X) = 3.12 – 1.57 = 1.55 bits/message

**ALGORITHM:**

1. Read the number of rows m and number of columns n for the joint probability matrix.
2. Read the individual matrix elements and display them.
3. Find out the summation of each row, which gives P (X0), P (X1), and P (X2).
4. Find out the summation of each column, which gives P (Y0), P (Y1), and P (Y2).
5. Find H (X). ****
6. Find H (Y). ****
7. Find H (X, Y). ****
8. Calculate H (X/Y) = H (X, Y) – H (Y) bits/message.
9. Calculate H (Y/X) = H (X, Y) – H (X) bits/message.
10. Display all the results.

**CODE:**

%noiseless channel

clc;

close all;

clear all;

disp ("CODE BY ISHAN MODI");

i=input("Enter the number of input : ");

q=input("Enter the joint probability matrix : ");

sum=0;

%probability P(x)

for n=1:i

w=0;

for m=1:i

p(n)=w+ q(n,m);

w=p(n);

end

end

disp('P(X):');

disp(p);

%Entropy H(x)

for n=1:i

H=sum+p(n)\*log(1/p(n))/log(2);

sum=H;

end

disp("H(X):");

disp(H);

%Conditional Probability matrix

for n=1:i

for m=1:i

a(n,m)=q(n,m)/p(n);

end

end

disp("P(Y/X):");

disp(a);

%Entropy H (y/x)

H1=0;

d=0;

for n=1:i

for m=1:i

if (a(n,m)>0)

Hl=d+(q(n,m)\*log2(1/(a(n,m))));

d=H1;

end

end

disp('H (Y/X) :');

disp(H1);

%Probability p (y)

for n=1:i

w=0;

for m=1:i

s(n)=w+q(m,n);

w=s(n);

end

end

disp('P (Y) :');

disp(s);

%entropy H(y)

k=0;

for n=1:i

H2=k+s(n)\*log2(1/s(n));

k=H2;

end

disp("H(Y):");

disp(H2);

%mutual information

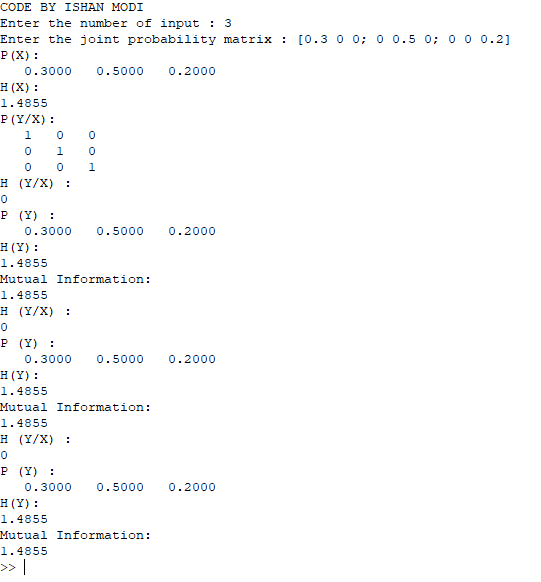
m=H2-H1;

disp("Mutual Information:");

disp(m);

end;

**OUTPUT:**



**CONCLUSION:** **Through this experiment we were able to understand the concept of Entropy and mutual information. We also understood the algorithm for computation of Entropy and mutual information and also wrote an OCTAVE code to understand the same.**

**ASSIGNMENT NO. 2**

**TITLE: Variable Length Source Coding: Encoding and Decoding of Shannon – Fano and Huffman coding**

**PROBLEM STATEMENT:** Write C/MATLAB program to implement algorithm for generation and evaluation of variable length source coding using

a) Shannon – Fano coding (Coding and Decoding)

b) Huffman Coding(Coding and Decoding)

c) Lempel Ziv dictionary technique

Compute entropy, average length and coding efficiency.

**OBJECTIVE:**

1. To understand the concept variable length source coding.
2. To implement algorithm for Huffman code and Shannon fano encoding
3. To compute entropy, average length and coding efficiency.

**THEORY:**

**Huffman coding:**

It is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table also called as code book, has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

Probability of occurrence is based on the frequency of occurrence of a data item. The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a *Code Book* which may be constructed for each block or a set of blocks. In all cases the code book plus encoded data must be transmitted to enable decoding.

The Huffman algorithm is now briefly summarized:

The simplest construction algorithm uses a priority queue where the node with lowest probability is given highest priority:

1. Create a leaf node for each symbol and add it to the priority queue.
2. While there is more than one node in the queue:
   1. Remove the two nodes of highest priority (lowest probability) from the queue
   2. Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
   3. Add the new node to the queue.
3. The remaining node is the root node and the tree is complete.

Since efficient priority queue data structures require O(log *n*) time per insertion, and a tree with *n* leaves has 2*n*−1 nodes, this algorithm operates in O(*n* log *n*) time, where *n* is the number of symbols.

If the symbols are sorted by probability, there is a linear-time (O(*n*)) method to create a Huffman tree using two queues, the first one containing the initial weights (along with pointers to the associated leaves), and combined weights (along with pointers to the trees) being put in the back of the second queue. This assures that the lowest weight is always kept at the front of one of the two queues:

1. Start with as many leaves as there are symbols.
2. Enqueue all leaf nodes into the first queue (by probability in increasing order so that the least likely item is in the head of the queue).
3. While there is more than one node in the queues:
   1. Dequeue the two nodes with the lowest weight by examining the fronts of both queues.
   2. Create a new internal node, with the two just-removed nodes as children (either node can be either child) and the sum of their weights as the new weight.
   3. Enqueue the new node into the rear of the second queue.
4. The remaining node is the root node; the tree has now been generated.

Although this algorithm may appear "faster" complexity-wise than the previous algorithm using a priority queue, this is not actually the case because the symbols need to be sorted by probability before-hand, a process that takes O(*n* log *n*) time in itself.

Many variations of Huffman coding exist, some of which use a Huffman-like algorithm, and others of which find optimal prefix codes.

1. *n*-ary Huffman coding
2. Adaptive Huffman coding
3. Huffman template algorithm
4. Length-limited Huffman coding/minimum variance huffman coding
5. Huffman coding with unequal letter costs.
6. Optimal alphabetic binary trees (Hu-Tucker coding)
7. The canonical Huffman code

Example: A source generates 4 different symbols with probability 0.4, 0.35, 0.2, 0.05. A binary tree is generated from left to right taking the two least probable symbols and putting them together to form another equivalent symbol having a probability that equals the sum of the two symbols. The process is repeated until there is just one symbol. The tree can then be read backwards, from right to left, assigning different bits to different branches. The final Huffman code is:

|  |  |
| --- | --- |
| **Symbol** | **Code** |
| a1 | 0 |
| a2 | 10 |
| a3 | 110 |
| a4 | 111 |

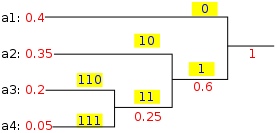
****

Fig. 2.1: Code Assignment

**Shannon Fano Coding:**

It is a technique for constructing a prefix code based on a set of symbols and their probabilities (estimated or measured). It is suboptimal in the sense that it does not achieve the lowest possible expected code word length like Huffman coding; however unlike Huffman coding, it does guarantee that all code word lengths are within one bit of their theoretical ideal

In Shannon–Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

The algorithm produces fairly efficient variable-length encodings; when the two smaller sets produced by a partitioning are in fact of equal probability, the one bit of information used to distinguish them is used most efficiently. Unfortunately, Shannon–Fano does not always produce optimal prefix codes; the set of probabilities {0.35, 0.17, 0.17, 0.16, 0.15} is an example of one that will be assigned non-optimal codes by Shannon–Fano coding.

For this reason, Shannon–Fano is almost never used; Huffman coding is almost as computationally simple and produces prefix codes that always achieve the lowest expected code word length, under the constraints that each symbol is represented by a code formed of an integral number of bits. This is a constraint that is often unneeded, since the codes will be packed end-to-end in long sequences.

**Example:** A source generates 7 different symbols with probability 0.1, 0.05, 0.2, 0.15, 0.15, 0.25, 0.1. Encode by Shannon fano method.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Character** | **Probability** | **Iter1** | **Iter2** | **Iter3** | **Iter4** | **Code** |
| **X6** | **0.25** | **1** | **1** |  |  | **11** |
| **X3** | **0.2** | **1** | **0** |  |  | **10** |
| **X4** | **0.15** | **0** | **1** | **1** |  | **011** |
| **X5** | **0.15** | **0** | **1** | **0** |  | **010** |
| **X1** | **0.1** | **0** | **0** | **1** |  | **001** |
| **X7** | **0.1** | **0** | **0** | **0** | **1** | **0001** |
| **X2** | **0.05** | **0** | **0** | **0** | **0** | **0000** |

**Algorithm:**

1. Declare all the input variable and array.
2. Take total number of symbols from the user (n).
3. Enter number of symbols should be less than or equal to 10.
4. If n s less then equal to zero, than ask the user to correct number of symbols.
5. Take probabilities for the symbol from user.
6. If sum of the probabilities is less than or greater than 1 ask the user to reentered the correct probabilities.
7. Sort the probabilities.
8. Form table for code generated.
9. Find value of entropy H (x).
10. Find average code word length and calculate efficiency.

**CoDE:**

clc;

clear all;

close all;

x=input('enter the number of symbols:'); %enter the number of symbols

for m=1:x

symbols(m)=input('enter the symbol no:'); %enter the symbol number

p(m)=input('enter the probability of symbol :'); %enter the probablity occurance

end

hx=0; %initialization of entropy

l=0; %initialization of efficiency

pkg load communications;

for m=1:x

dict=huffmandict(symbols,p); %function to create dictionary

hcode=huffmanenco(m,dict); %encoding

dsig=huffmandeco(hcode,dict); %decoding

code\_length(m)=length(hcode); %to find length

hx=hx-(p(m)\*log2(p(m))); %formula for entropy

l=l+code\_length(m)\*p(m);

end

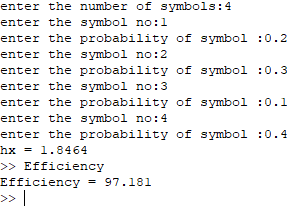
allencode=huffmanenco(symbols,dict); %ecoding code

decodeall=huffmandeco(allencode,dict); %decoding code

display(hx); %display efficiency value

efficiency=(hx/l)\*100; %formula for efficiency

**OUTPUT:**



**Conclusion:** T**hrough this experiment we were able to understand concept of variable length source coding. we also implemented algorithm for huffman code and computed entropy, average length and coding efficiency.**

**QUESTIONS:**

1. What is a prefix code?
2. Explain Kraft inequality?
3. What is the efficiency of any code?
4. What are the steps for Shannon-fano encoding mechanism?
5. What is run length encoding?
6. Comment on the efficiency of Shannon-Fano coding method.
7. Explain the steps for Lempel-Ziv algorithm.
8. Compare Lempel-Ziv & Huffman Encoding mechanism.
9. Distinguish between Lossy &Lossless data compression with examples.

**ASSIGNMENT NO. 3**

**TITLE: Linear Block Code: Encoding and Decoding.**

**PROBLEM STATEMENT:** Write MATLAB program to implement the algorithms for generation and decoding of linear block code.

**OBJECTIVE:**

1. To implement the systematic encoding of linear block code using generating matrix.
2. To implement the systematic decoding of linear block code using Parity check matrix.

**THEORY:**

Shannon Demonstrated that, by proper encoding of the information; errors induced by a noisy channel or storage medium can be reduced to any desired level, without sacrificing the rate of information transmission or storage.

Reliable transmission of information over noisy channels requires the use of error correcting codes which encode input in such a way that errors can be detected and corrected at the receiving site.

The basic idea behind error correcting codes is an addition of some controlled redundancy in the form of extra symbol to a message prior to transmission of message through a noisy channel. This redundancy is added in a controlled manner. The encoded message when transmitted might be corrupted by noise in the channel. At the receiver, the original message can be recovered from the corrupted one if the errors are within the limit for which the code has been designed.

1. Error correcting capability in terms of the number of errors that it can rectify.
2. Fast and efficient encoding of the message.
3. Fast and efficient encoding of the received message.
4. Maximum transfer of information bits per unit time (i.e., fewer overheads in terms of redundancy).

**Linear Block Codes**

An (n, k) **block code** *C* over an alphabet of *q*symbols is a set of n-vectors called **codewords** or **code vectors.** Associated with the code is an encoderwhich maps a message, a k- tuple , to its associated codeword.

A block code *C* over a field, can also be defined as, of *q* symbols of length n and codewords is a *q-ary* linear (n, *k)*code if and only if its codewords form a k-dimensional vector subspace of the vector space of all the n-tuples . The number n is said to be the lengthof the code and the number k is the dimensionof the code. The rate ofthe code is ***R*** = k/n.

**CODE RATE:**

The code rate of an (n,k) code is defined as the ratio (k/n) and denotes the fraction of the codeword that consists of information symbols.

**MINIMUM DISTANCE:**

The minimum distance of a code is the minimum distance between any two code words.

**MINIMUM WEIGHT:**

The minimum weight of a code is the smallest weight of any non-zero element codeword and is denoted by w.

### Constraint on and :

**Let** given  linear code,

Where block length

dimension of code words

hamming distance

1. Block length :

This parameter gives us set of vectors which can be used as code words.

1. Dimension :

This is based on some logic by which we select some of vectors from  available vectors. Number of vectors which can be called as code words is  and  is called adimension of code word.

1. Hamming distance :

The (Hamming) distance between and., is defined as the number of components in which they differ; i.e., if





Then  = 5

It is always desirable to have higher value of , because higher value of  provides us easy decoding. With increase in distance, there is decrease in the number of code word. Therefore the necessary and sufficient condition for  and  is that the value of and should always high, but it is not possible practically. So, the compromise between  and  is made.

**GENERATOR MATRIX:**

The generator matrix is a matrix having k rows and n columns i.e., it is a k\*n matrix with rank k. Since the choice of the basis vectors is not unique, the generator matrix is not unique for a given linear code. The generator matrix is of the following form:



where,  = identity matrix *P =* Parity matrix

*k* = Original message length *n* = Code vector length

**PROPERTIES OF PARITY MATRIX:**

1. No raw parity matrix contains all non-zero elements.
2. Each row of polarity matrix is unique.

**ERROR DETECTION CAPACITY:**

Minimum error detection capability is denoted by s and is given by,



where, = Minimum distance

s = Error detection capability

**ERROR CORRECTION CAPABILITY:**

Minimum error correction capability is denoted by t, and is given by,



where, = Minimum distance, t = Error correction capability

**PARITY CHECK MATRIX:**

It is possible to detect a valid code word and such a matrix is called the parity check matrix denoted by *H*. For decoding purpose, we consider it’s transpose .  is of the following form, 

There are different formats of transpose of parity check matrix depending upon generator matrix (G).

if,  then, 

if,  then, 

if,  then, 

**Algorithm:**

* 1. Input the values of n and k.
  2. Input the parity matrix.
  3. Calculate generator matrix G.
  4. Input the data matrix. M [i.e. any one combination out of the total possible combinations.]
  5. Calculate the code using X = M \* G.
  6. In the same way calculate all the code words.
  7. Introduce error in the m bit position changing it either from 1 to 0 or 0 to 1.
  8. Calculate the syndrome.
  9. Compare error pattern with corresponding syndrome
  10. Evaluate correct code Xc

**CODE FOR ENCODING:**

clc;

clear all;

n=input('Enter value of n=');

k=input('Enter value of k=');

m=n-k;

P=input('Enter parity matrix=');

disp(P);

I=eye(k,k);

disp('Generator Matrix G=');

G=[I P];

disp(G);

d=0:(2^k)-1;

u=dec2bin(d,3);

disp('Display all possible messages=');

disp(u);

cws=u\*G;

cw=rem(cws,2);

disp('Generated all codewords for the messages are=');

disp(cw);

B=input('Enter the messages=');

disp('entered message is=');

disp(B);

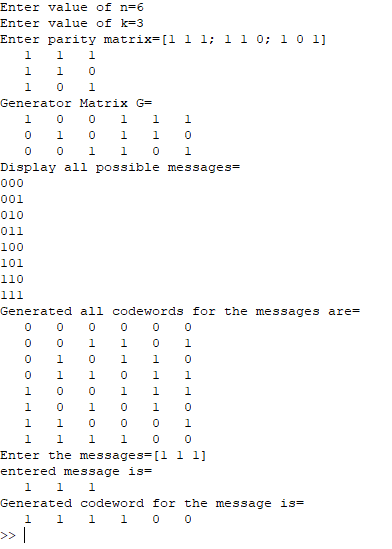
C=B\*G;

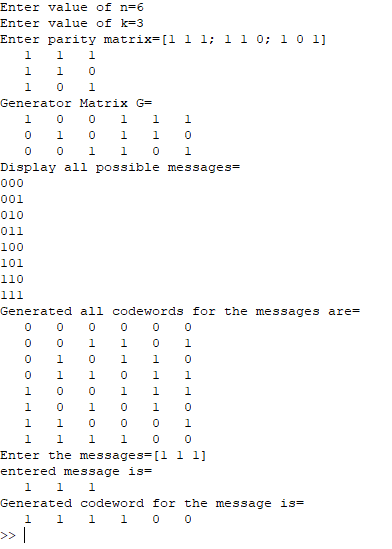
Code\_C=rem(C,2);

disp('Generated codeword for the message is=');

disp(Code\_C);

**OUTPUT FOR ENCODING:**





**CODE FOR DECODING:**

clc;

clear all;

close all;

n=input('Enter the codeword length,n= ');

k=input('Enter the no of message length of the codeword,k= ');

P=input('Enter the coeff of parity matrix :');

r=input('Enter the received codeword');

a=P' %Transpose of parity matrix,P

I=eye(n-k) %Identity matrix of the order of (n-k)

H=cat(2,a,I) %Parity Check Matrix H

TransposeH=H'

s=r\*TransposeH %Syndrome vector,s=received codeword \* H'

l=rem(s,2)

if( l==0)

disp('no need of correction');

else

e=eye(n) %returns the n-by-n identity matrix.

el=e\*TransposeH

sl=rem(el,2)

for i=1:n

if l==sl(i,:)

break;

end

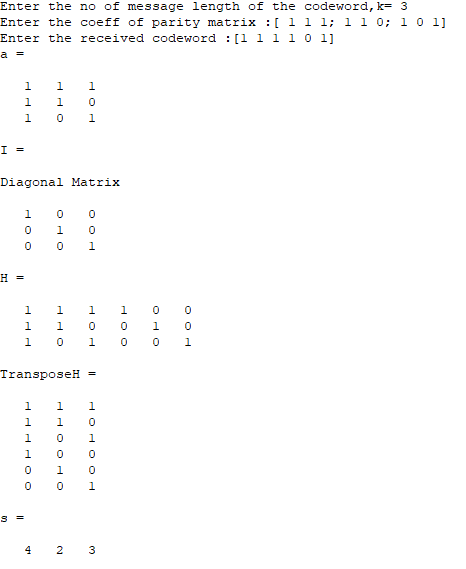
end

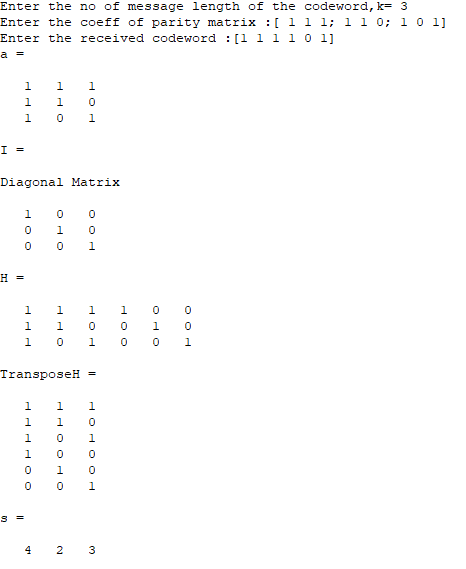
correctedCodeword=xor(r,e(i,:)) %correctedCodeword=r xor e

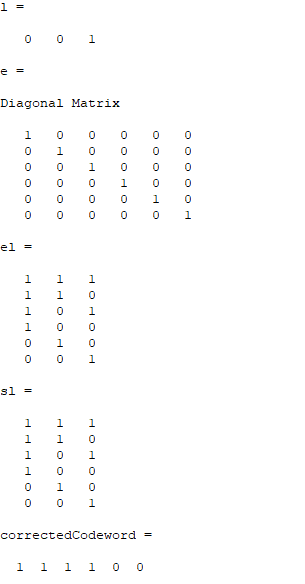
mes=correctedCodeword(1:k)

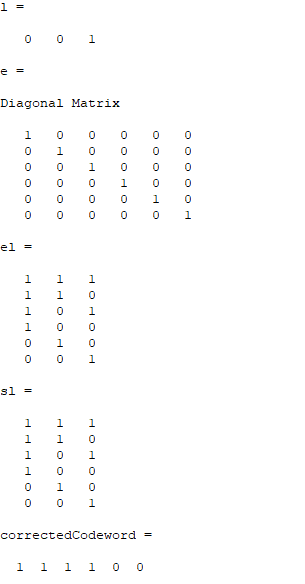
end

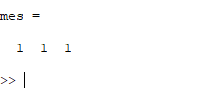
**OUTPUT FOR DECODING:**











**CONCLUSION:** **Through this experiment we implemented the systematic encoding of linear block code using generating matrix. We also implemented the systematic decoding of linear block code using Parity check matrix.**

**QUESTIONS:**

1. What do you mean by hamming distance?
2. How to obtain minimum hamming distance?
3. How to obtain error Detecting capability
4. How to obtain error correcting capability

**ASSIGNMENT NO. 4**

**TITLE: Cyclic Code: Encoding and Decoding.**

**PROBLEM STATEMENT:** Write MATLAB program to implement the algorithms for generation and decoding of cyclic code.

**OBJECTIVE:**

1. To implement the systematic encoding of Cyclic Code.
2. To implement the systematic decoding of Cyclic Code.

**THEORY:**

Given a vector,the vector

is said to be a cyclic shift of c to the right. A shift by r places to the right produces the vector 

Definition : An (n, k) block code C is said to be cyclic if it is linear and if for every codeword in in **C**, its right cyclic shift is also in C.

**Encoding of cyclic code**

Encoding Rule to generate Codeword from generator polynomial



Where  is information polynomial and let 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message | Information Polynomial | Code poly | Code poly | Code word |
| 0000 | 0 |  | 0 | 0000000 |
| 0001 |  |  |  | 0001101 |
| 0010 |  |  |  | 0011010 |
| 0011 |  |  |  | 0010111 |
| 0100 |  |  |  | 0110100 |
| 0101 |  |  |  | 0111001 |
| 0110 |  |  |  | 1001110 |
| 0111 |  |  |  | 0100011 |
| 1000 | 1 | 1 |  | 1101000 |
| 1001 |  |  |  | 1100101 |
| 1010 |  |  |  | 1110010 |
| 1011 |  |  |  | 1111111 |
| 1100 |  |  |  | 1011100 |
| 1101 |  |  |  | 1010001 |
| 1110 |  |  |  |  |
| 1111 |  |  |  |  |

**Division Algorithm for cyclic code**

Let 





For systematic encoding



**Decoding Rule to decode cyclic code**

Let Received word at receiver is 

Syndrome polynomial is given by





**Problem:**

Obtain systematic (7,4) Cyclic code for generating polynomial 

**CoDE:**

%%% cyclic code

clc; clear all;

pkg load communications;

#{

adds named package 'communication package' here to the path ,

after loading package it is possible to use functions provided by the package

#}

n=input('enter the length of code word:')

k=input('enter the length of msg word:')

m=input('enter the msg word:')

g=cyclpoly(n,k,'max')%https://octave.sourceforge.io/communications/function/cyclpoly.html

gx=polyout(g,'x') %https://octave.sourceforge.io/octave/function/polyout.html

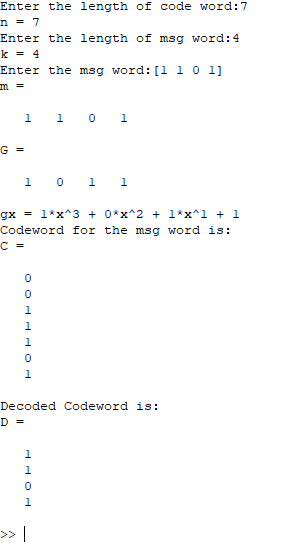
disp('codeword for the msg word is:')

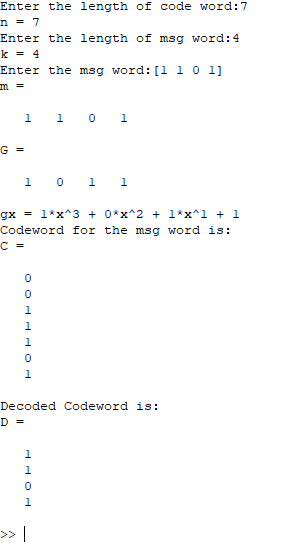
c=encode(m,n,k,'cyclic',g)%%https://octave.sourceforge.io/communications/function/encode.html

disp('decoded codeword is:')

d=decode(c,n,k,'cyclic',g)

**OUTPUT:**





**Conclusion:** **Through this experiment we were able to implement the systematic encoding and decoding of cyclic code.**

**QUESTIONS:**

* 1. What are important properties of Cyclic code
  2. What are properties of syndrome table
  3. Why cyclic code are more suitable for burst errors.
  4. Draw a circuit implementation of cyclic code (both encoding and decoding).

**ASSIGNMENT NO. 5**

**TITLE: Convolution Code.**

**PROBLEM STATEMENT:** Write MATLAB program to implement the algorithms for generation Convolution Code by

a. Code Tree.

b. Code Trellis.

**OBJECTIVE:**

1. To implement the encoding of Convolution Code by
   1. Code Tree.
   2. Code Trellis.

**THEORY:**

**Convolutional Codes**

In block coding, the encoder accepts *k*-bit message block and generates an n-bit code word. i.e. code words are produced on a block-by-block basis. We know that we have serial and parallel communication, as far as serial data is concerned, provisions must be made in the encoder to buffer an entire message block before generating the associated code word. Thus buffering introduces delay and hence when data / message bits come in serially, buffer is undesirable. In such situation use of convolutional coding may be preferred method.

In convolutional coding the current information frame with previous *m* information frames are used to obtain a single codeword frame.

**Information frame**

Smaller blocks of uncoded data of length k0 are used for encoding purpose. Theses are called Information frame

Thus convolutional coding implies that encoders have *memory,* which retain the previous *m* incoming information frames. The codes that are obtained in this fashion are called Tree Codes. An important subclass of tree codes, used frequently in practice, is called Convolutional Codes.

**Tree codes and Trellis codes**

We assume that we have an infinitely long stream of incoming symbols. This stream of symbols is first broken up into (segments of *k0* symbols. Each segment is called an) information frame

The encoder consists of two parts

1. Memory – basically a shift register
2. a logic circuit.

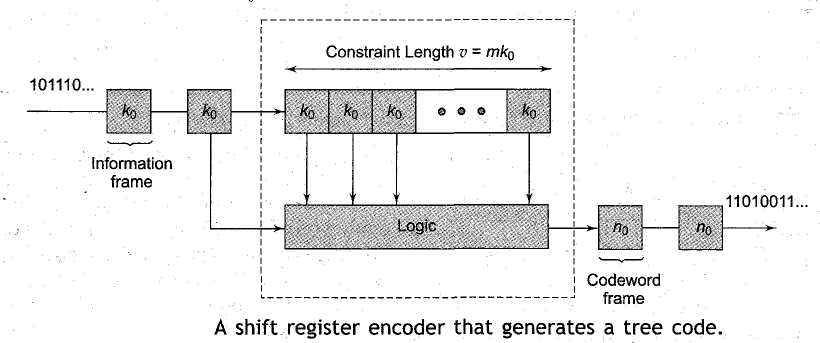


Fig. 5.1: A shift register encoder that generates a tree code

The memory of the encoder can store *m* information frames. Each time a new information frame arrives, it is shifted into the shift register and the oldest information frame is discarded. At the end of any frame time the encoder has *m* most recent information frames in its memory, which corresponds to a total of *mk0* information symbols.

**Constraint Length of a shift register encoder:**

It is defined as the number of symbols can be stored in memory of shift register.

If the shift register encoder stores *m* previous information frames of-length k0the constraint length of this encoder = *v* = mk0

**Formal Definition of Tree Code:**

It is a mapping, from the set of semi infinite sequences of elements of *GF(q)* into itself such that, if for any M, two semi infinite sequences agree in the first Mk0 components, then their images agree in the first Mn0 components.

Wordlength of a shift register encoder: It is defined as 

The Blocklength of a shift register encoder is defined as



Code rate



**Convolutional code:**

A (n0 ,k0)tree code that is linear, time invariant, and has a finite word length  is called an *(n, k)* Convolutional Code.

**Sliding Block Code**

A (n0 ,k0) tree code that is time-invariant and has a finite word length k is called (n,k) sliding block code.

A linear sliding block code is called convolutional code.

Consider a convolution code shown in figure below.

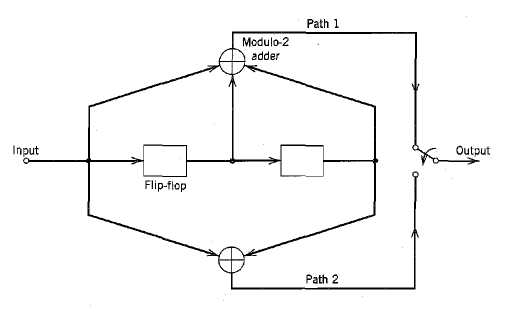


Fig. 5.2:Convolutional encoder with *n* = 2 and *k =* 3.

Code rate =.

The encoder operates on the incoming message sequence, one bit at a time and is *nonsystematic* codes.

Each path connecting the output to the input of a convolutional encoder may be characterized in terms of its *impulse response,* defined as the response of that path to a symbol 1 applied to its input, with each flip-flop in the encoder set initially in the zero state.

Equivalently, we may characterize each path in terms of a *generator polynomial,* defined as the *unit-delay transform* of the impulse response.

let the *generator* sequence denote the impulse response of the ith path, where the coefficientsequal 0 or 1. Correspondingly, the *generator polynomial* of the ith path is defined by



where *D* denotes the unit-delay variable.

**Working:**

The convolutional encoder of above Figure, has two paths numbered 1 and 2. The impulse response of path 1 (i.e., upper path) is (1, 1,1). Hence, the corresponding generator polynomial is given by



Similarly for path 2 (lower path) is (1,0,1)



For the message sequence (10011), say, we have the polynomial representation

m(D) = 1 + D3 + D4

We know from Fourier transformation, convolution in the time domain is transformed into multi­plication in the D-domain.

Hence, the output polynomial of path 1 is given by



We can deduce that the output sequence of path 1 is (1111001)

Similarly for path 2



The output sequence of path 2 is therefore (1011111).

Finally, multiplexing the two output sequences of paths 1 and 2, we get the encoded sequence

c = (11, 10, 11, 11, 01, 01, 11)

Note:

1. The message sequence of length *L = 5* bits produces an encoded sequence of length *n{L*+*K-*1) = 14 bits.
2. For the shift register to be restored to its zero initial state, a terminating sequence of *K -*1=2 zeros is appended to the last input bit of the message sequence. The terminating sequence of *K —* 1 zeros is called the *tail of the message.*

**TREE CODE:-**

Each branch of the tree represents an input symbols with the corresponding pair of output binary symbols indicated on the branch. The convolution used to distinguish the input binary symbols 0 & 1 is as follows. An input specifies path the upper branch of a bifurcation, whereas input specifies the lower branch. A specific path in the tree is traced from left to right in accordance with the input sequence. The corresponding coded symbols on the branches of that path constitute the input sequence. The tree becomes repetitive after certain number of branches where the number of branches is associated with the memory of the encoder.

**TRELLIS CODE:-**

The tree code can be collapsed into a new form called TRELLIS.

Trellis diagram are messy but generally preferred over both the tree and the state diagrams because they represent linear time sequencing of events to produce the trellis diagrams, advantage is taken of the fact that the tree structure repeats itself after ‘K’ branches that is it is periodic with period ‘K’. The x axis is discrete time and all possible states are shown on the y axis. Trellis moves horizontally with the passage of time. Each transition means new bit have arrived.

Each state is connected to the next state by the allowable codeword for that state. There are only two choices possible at each state. These are only determined by the arrived of either as ‘O’ bit or ‘1’ bit. The arrows show the input bit and output bits are shown in parenthesis. The arrows going upwards unique to each case, same as both the state and tree diagram are trellis can be drawn for as many periods as desired. Each period repeats the possible transitions one time interval section of a fully formed encoding trellis structure completely defines this code. Some sections can be shown for viewing a code symbol sequence as a function of time.

Steps for Code Tree Implementation:

1. Tree becomes repetitive after 3rd branch. Beyond the 3rd branch, the two nodes labeled are identical nodes.
2. The encoder has memory M = K-l =2 message bits. Hence when third message bit enters the encoder, the 1st message bit is shifted out of the register.
3. In the code tree starting with 1 & 0, if there is ' 1' in the input sequence, then proceed downward. (This is shown by dotted line) & note down the correct code written on that line.
4. If there is '0' in the input sequence, then go upward (shown by solid line) and note down code written on that line.
5. Thus trace the code - tree up to level equal to number of bits in input sequence to get the corresponding output sequence.

Step for Code Trellis implementation:

1. If there is '0' in k0, then trace upward [i.e. solid line & note down code written above the line.
2. If there is ' 1' in k0, then trace downward [i.e. dotted line & note down code written above the line

Thus for k0 = l 1 0 1 00 0

We get n0 =11010100101100

**VITERBI’S ALGORITHM:**

Lets represent the received signal by ‘y’. Convolution decoding operates continuously on input data. Let 1 and 0 have same transmission error probability. Then matric in the discrepancy between the received signal ‘y’ and decoded signal at a particular mode. This matric can be added over a few nodes for a particular path.

In this method output or the received code is compared with the trellis diagram. If the output of the node in the trellis diagram matches with the received code then all paths are checked and the respective matric are written down.

In case if there are two paths having same matric then only one of them is continued. Otherwise the path having lowest matric is chosen whenever the path is broken it shows the message bit m=1 and if it is continuous message bit then m=0.If it is continuous between two nodes method of decoding in viterbi algorithm is called max likelihood decoding.

In this decoding,

Surviving path=2(k-1)R

Where,

K=constant length

R=message bit

If the number of message bits cleared decoded are very large then storage requirement is also large since the decoder has to store multiple paths. To avoid this, matric diversion effect is used.

**PROBLEM :**

For the convolution encoder shown draw the trellis diagram and using viterbi algorithm decode the sequence 1 1 1 0 1 1 1 1 0 1 0 1 1 1

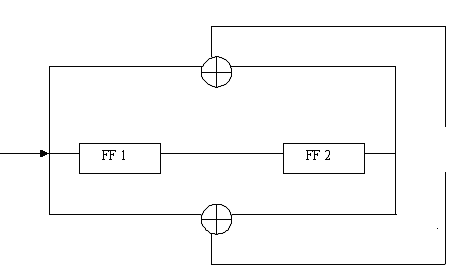
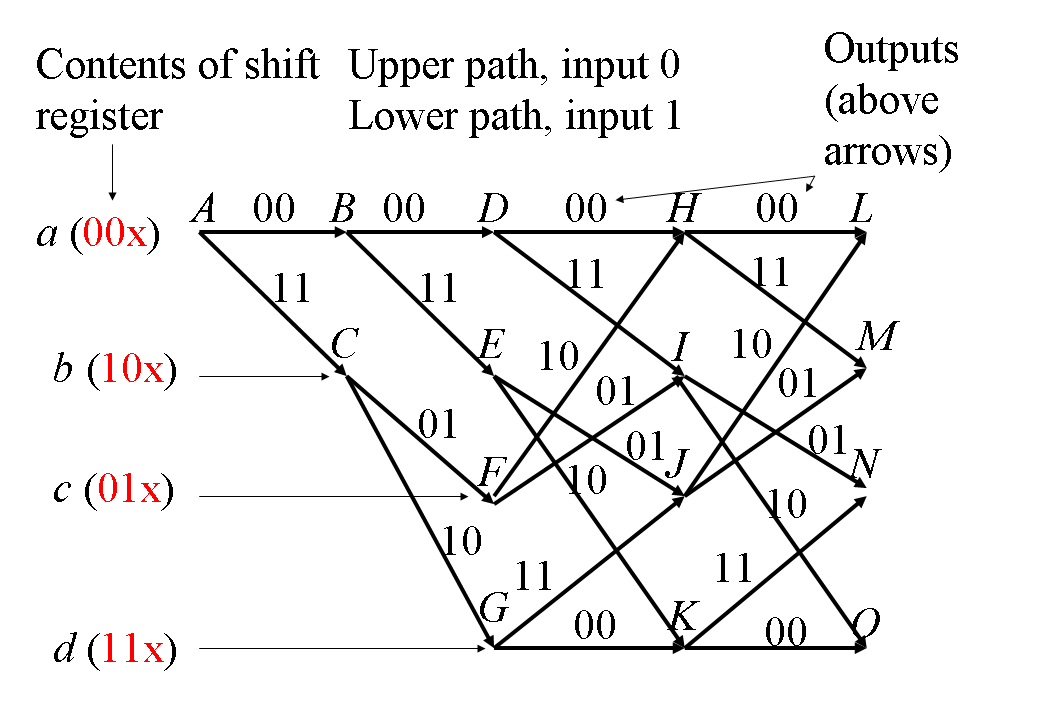


Fig. 6.1: Convolutional Encoder

GIVEN SEQUENCE: 01000100

The path with least metric is the 1st path with metric 4.hence it is surviving path and it gives message bit output. Hence the output is 1001100.

**TRELLIS DIAGRAM**



**CODE:**

clc;

clear all;

pkg load communications;

k=input('input constraint length : '); %Enter the length of the constraint

g=input('enter coefficients of generator paths : ');

msg=input('enter message : ');

t=poly2trellis(k,g);

chcode=convenc(msg,t)

disp('encoded sequence is : ')

disp(chcode);

%addition of random error

err=randerr(1,length(chcode),2);

disp('error pattern is : ');

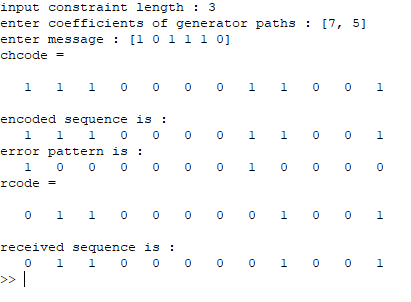
disp(err)

rcode=rem(chcode+err,2)

disp('received sequence is : ')

disp(rcode);

**OUTPUT:**



**CONCLUSION: Through this experiment we were able to implement the encoding of Convolution Code by Code Tree and Code Trellis.**

**QUESTIONS:**

1. Write polynomial description of convolutional code
2. For convolutional code, how to obtain dfree
3. With neat example encoding schemes for convolutional code. Draw circuit, state table, state diagram and trellis diagram.
4. Explain sequential decoding scheme for convolutional code
5. Explain how decoding of convolutional code can be achieved by viterbi decoding
6. Explain concept of trace-back length.

**ASSIGNMENT NO. 6**

**TITLE: BCH CODE ENCODING AND DECODING.**

**PROBLEM STATEMENT:** Write MATLAB program to implement the algorithms for encoding and decoding of BCH Code.

**OBJECTIVE:**

To implement the encoding and decoding of BCH Code.

**THEORY:**

In coding theory the BCH codes form a class of parameterized error correcting code BCH codes were invented in 1959 by [Hocquenghem](http://en.wikipedia.org/wiki/Alexis_Hocquenghem), and independently in 1960 by [Bose](http://en.wikipedia.org/wiki/Raj_Chandra_Bose) and [Ray-Chaudhuri](http://en.wikipedia.org/wiki/D.K._Ray-Chaudhuri). The acronym *BCH* comprises the initials of these inventors' names.

The principal advantage of BCH codes is the ease with which they can be decoded, via an elegant [algebraic](http://en.wikipedia.org/wiki/Abstract_algebra) method known as syndrome decoding. This allows very simple electronic hardware to perform the task, obviating the need for a computer, and meaning that a decoding device may be made small and low-powered. As a class of codes, they are also highly flexible, allowing control over block length and acceptable error thresholds, meaning that a custom code can be designed to a given specification (subject to mathematical constraints). [Reed–Solomon codes](http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction), which are BCH codes, are used in applications such as satellite communications, [compact disc](http://en.wikipedia.org/wiki/Compact_disc) players, [DVDs](http://en.wikipedia.org/wiki/DVD), [disk drives](http://en.wikipedia.org/wiki/Disk_storage), and [two-dimensional bar codes](http://en.wikipedia.org/wiki/Bar_codes).

In technical terms a BCH code is a multilevel [cyclic](http://en.wikipedia.org/wiki/Cyclic_code) variable-length [digital](http://en.wikipedia.org/wiki/Digital) error-correcting code used to correct multiple random error patterns. BCH codes may also be used with multilevel [phase-shift keying](http://en.wikipedia.org/wiki/Phase-shift_keying) whenever the number of levels is a [prime number](http://en.wikipedia.org/wiki/Prime_number) or a power of a prime number. A BCH code in 11 levels has been used to represent the 10 decimal digits plus a sign [digit](http://en.wikipedia.org/wiki/Numerical_digit)

**Construction:**

A BCH code is a [polynomial code](http://en.wikipedia.org/wiki/Polynomial_code) over a [finite field](http://en.wikipedia.org/wiki/Finite_field) with a particularly chosen [generator polynomial](http://en.wikipedia.org/wiki/Polynomial_code). It is also a [cyclic code](http://en.wikipedia.org/wiki/Cyclic_code).

**Simplified BCH codes:**

Fix a [finite field](http://en.wikipedia.org/wiki/Finite_field) *GF*(*qm*), where *q* is a prime. Also fix positive integers *n*, and *d* such that *n* = *qm* − 1 and . We will construct a [polynomial code](http://en.wikipedia.org/wiki/Polynomial_code) over *GF*(*q*) with code length *n*, whose minimum [Hamming distance](http://en.wikipedia.org/wiki/Hamming_distance) is at least *d*. What remains to be specified is the generator polynomial of this code.

Let α be a [primitive *n*th root of unity](http://en.wikipedia.org/wiki/Primitive_nth_root_of_unity) in *GF*(*qm*). For all *i*, let *mi*(*x*) be the [minimal polynomial](http://en.wikipedia.org/wiki/Minimal_polynomial_(field_theory)) of α*i* with coefficients in *GF*(*q*). The generator polynomial of the BCH code is defined as the [least common multiple](http://en.wikipedia.org/wiki/Least_common_multiple)

Example:

Let *q* = 2 and *m* = 4 (therefore *n* = 15). We will consider different values of *d*. There is a primitive root satisfying α4 + α + 1 = 0 its minimal polynomial over *GF*(2) is :*m*1(*x*) = *x*4 + *x* + 1.

Note that in *GF*(24), the equation (*a* + *b*)2 = *a*2 + 2*ab* + *b*2 = *a*2 + *b*2 holds, and therefore *m*1(α2) = *m*1(α)2 = 0. Thus α2 is a root of *m*1(*x*), and therefore *m*2(*x*) = *m*1(*x*) = *x*4 + *x* + 1.

To compute *m*3(*x*), notice that, by repeated application of (1), we have the following linear relations

Five right-hand-sides of length four must be linearly dependent, and indeed we find a linear dependency α12 + α9 + α6 + α3 + 1 = 0. Since there is no smaller degree dependency, the minimal polynomial of α3 is :*m*3(*x*) = *x*4 + *x*3 + *x*2 + *x* + 1. Continuing in a similar manner, we find

The BCH code with *d* = 1,2,3 has generator polynomial

It has minimal Hamming distance at least 3 and corrects up to 1 error. Since the generator polynomial is of degree 4, this code has 11 data bits and 4 checksum bits.

The BCH code with *d* = 4,5 has generator polynomial

It has minimal Hamming distance at least 5 and corrects up to 2 errors. Since the generator polynomial is of degree 8, this code has 7 data bits and 8 checksum bits.

The BCH code with *d* = 6,7 has generator polynomial

It has minimal Hamming distance at least 7 and corrects up to 3 errors. This code has 5 data bits and 10 checksum bits.

**Decoding:**

There are many algorithms for decoding BCH codes. The most common ones follow this general outline:

1. Calculate the syndrome values for the received vector
2. Calculate the error locator polynomials
3. Calculate the roots of this polynomial to get error location positions.
4. Calculate the error values at these error locations.

**Calculate the syndromes**

The received vector *R* is the sum of the correct codeword *C* and an unknown error vector *E*. The syndrome values are formed by considering *R* as a polynomial and evaluating it at \alpha^c,\ldots,\alpha^{c+d-2}. Thus the syndromes are[[3]](http://en.wikipedia.org/wiki/BCH_code#cite_note-2)

*sj* = *R*(α*c* + *j* − 1) = *C*(α*c* + *j* − 1) + *E*(α*c* + *j* − 1)

for *j* = 1 to *d* − 1. Since α*c* + *j* − 1 are the zeros of *g*(*x*), of which *C*(*x*) is a multiple, *C*(α*c* + *j* − 1) = 0. Examining the syndrome values thus isolates the error vector so we can begin to solve for it.

If there is no error, *sj* = 0 for all *j*. If the syndromes are all zero, then the decoding is done.

**Calculate the error location polynomial**

If there are nonzero syndromes, then there are errors. The decoder needs to figure out how many errors and the location of those errors.

If there is a single error, write this as E(x) = e\,x^i, where *i* is the location of the error and *e* is its magnitude. Then the first two syndromes are

s_1 = e\,\alpha^{c\,i}

s_2 = e\,\alpha^{(c+1)\,i} = \alpha^i s_1

so together they allow us to calculate *e* and provide some information about *i* (completely determining it in the case of Reed-Solomon codes).

If there are two or more errors,

E(x) = e_1 x^{i_1} + e_2 x^{i_2} + \ldots

It is not immediately obvious how to begin solving the resulting syndromes for the unknowns *ek* and *ik*.

**Peterson Gorenstein Zierler algorithm**

Peterson's algorithm is the step 2 of the generalized BCH decoding procedure. We use Peterson's algorithm to calculate the error locator polynomial coefficients  \lambda_1 , \lambda_2, \dots, \lambda_{t} of a polynomial  \Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_{t}x^{t} 

Now the procedure of the Peterson Gorenstein Zierler algorithm for a given (*n*,*k*,*dmin*) BCH code designed to correct [t=\frac{d_{min}-1}{2}]errors is

* First generate the Matrix of 2*t* syndromes
* Next generate the S_{t\times t}matrix with elements that are syndrome values

S_{t \times t}=\begin{bmatrix}s_1&s_2&s_3&\dots&s_t\\
s_2&s_3&s_4&\dots&s_{t+1}\\
s_3&s_4&s_5&\dots&s_{t+2}\\
\dots&\dots&\dots&\dots&\dots\\
s_t&s_{t+1}&s_{t+2}&\dots&s_{2t-1}\end{bmatrix}

* Generate a *ctx*1 matrix with elements

C_{t \times 1}=\begin{bmatrix}s_{t+1}\\
s_{t+2}\\
\vdots\\
\vdots\\
s_{2t}\end{bmatrix}


* Let Λ denote the unknown polynomial coefficients, which are given by

\Lambda_{t \times 1} = \begin{bmatrix}\lambda_{t}\\
\lambda_{t-1}\\
\vdots\\
\lambda_{2}\\
\lambda_{1}\end{bmatrix}


* Form the matrix equation

S_{t \times t} \Lambda_{t \times 1}  = C_{t \times 1\,} 

* If the determinant of matrix S_{t \times t}exists, then we can actually find an inverse of this matrix and solve for the values of unknown Λ values.
* If  \det(S_{t \times t}) = 0 , then follow

if *t* = 0

then declare an empty error locator polynomial

stop Peterson procedure.

end

set  t \leftarrow t -1

continue from the beginning of Peterson's decoding

* After you have values of Λ you have with you the error locator polynomial.
* Stop Peterson procedure.

**Problem**:

Consider (15,7) double error correcting BCH code with g(x)=x8+x7+x6+x4+1.

If m(x)=(1+x),the transmitted code i.e m=[0000011],c(x)=m(x).g(x)

c(x)=x9+x6+x5+x4+x+1

c=[1101110111],r(x)=x9+x8+x6+x5+x4+x3+x+1

i.e we introduced errors in 3rd & 8th bit., r=[1101110111], correct the errors.

**CODE:**

clc; clear all;

pkg load communications;

m =input('Enter degree of generator poly-> ');

n = 2^m -1;

k = input('Enter length of msg -> ');

t = input('Enter error correcting capability -> ');

msg = input('Enter message(each of length k bits) -> ');

printf("\n");

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BCH encoding \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

code = bchenco(msg, n, k);

disp('bch encoded message is ');

disp(code);

printf("\n");

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* noisy \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

noisy = mod(randerr(size(msg),n) + code,2);

disp('received message (after error)is ');

disp(noisy);

printf("\n");

disp('\*\*\*\*\*\*\*\*\*\*\*\*\* BCH decoding \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

[dec err] = bchdeco(noisy, k, t);

disp('after decoding, obtained msg is ');

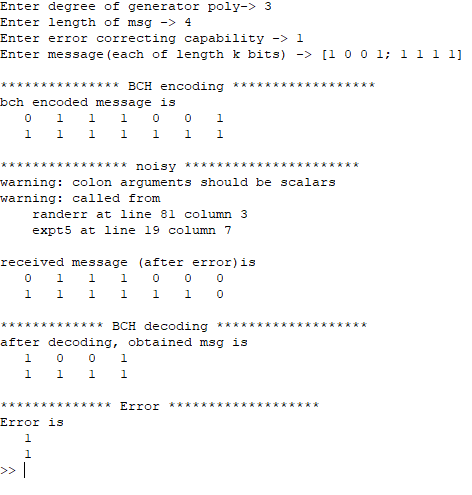
disp(dec);

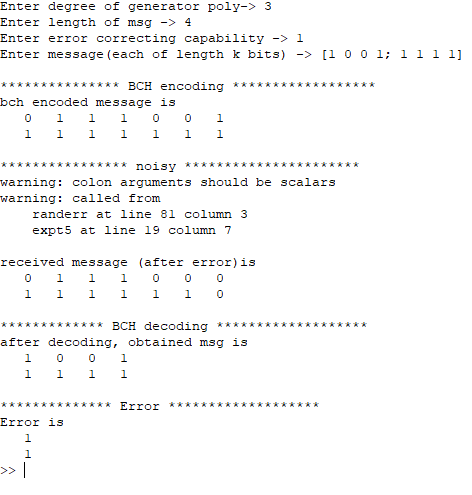
printf("\n");

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\* Error \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('Error is ');

disp(err);





**CONCLUSION:** **Through this experiment we were able to implement the encoding and decoding of BCH Code.**

**QUESTIONS:**

1. Obtain elements of field whose primitive polynomial is 
2. What do you mean by burst errors.
3. Write decoding scheme to correct burst errors.

**ASSIGNMENT NO. 7**

**TITLE: Implementation of ARQ Technique**

**PROBLEM STATEMENT:** Write a simulation program to implement ARQ techniques**.**

**OBJECTIVE:**

To study implementation of ARQ algorithm

**THEORY:**

Automatic Repeat request (ARQ), also known as Automatic Repeat Query, is an error-control method for data transmission that uses acknowledgements (messages sent by the receiver indicating that ithas message has correctly received a data frame or packet) and time outs (specified period so f time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable service.

If the sender does not receive an acknowledgment before the timeout, it usua**l**y re-transmits the frame/packet until the sender receives an acknowledgment or exceeds a predefined number of re-transmissions.

The types of ARQ protocols include Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ/Selective Reject. A**l** three protocols usually use some form of sliding window protocol to te**l** the transmitter to determine which (if any) packets need to be retransmitted. These protocols reside in the Data Link or Transport Layers of the OSI model.

A number of patents exist for the use of ARQ in live video contribution environments. In these high throughput environments negative acknowledgements are used to drive down overheads.

**ALGORITHM:**

Step1.Start the Program

Step2.Generate a random that gives the total number of frames to be transmitted

Step3.Transmit the first frame

Step4.Receive the Acknowledgement for the first frame

Step5.Transmit the next frame

Step6.Find the remaining frame to be sent

Step7.If an Acknowledgement is not received for a particular frame, retransmit that frame alone again

Step8.Repeat step 5 to7 tillnumber of remaining frames to be send becomes zero

Step9. Stop the program

**CODE:**

#include <stdio.h>

#include<stdlib.h>

void main()

{

int i,n,r,a;

n=5;

printf("The number of packets are:%d\n",n);

for(i=1;i<=n;i++)

{

printf("The packet sent is:%d\n",i);

r=rand()%2;

if (r==1)

{

a=rand() %2;

if (a==1)

{

printf("Aknowledgement number:%d\n",i+1);

}

else

{

printf("No acknowledgement number:%d\n",i+1);

i--;

}

}

else

{

printf("time out,resend\n");

i--;

}

}

}

**OUTPUT:**

The number of packets are:5

The packet sent is:1

No acknowledgement number:2

The packet sent is:1

Aknowledgement number:2

The packet sent is:2

Aknowledgement number:3

The packet sent is:3

time out,resend

The packet sent is:3

time out,resend

The packet sent is:3

Aknowledgement number:4

The packet sent is:4

time out,resend

The packet sent is:4

No acknowledgement number:5

The packet sent is:4

Aknowledgement number:5

The packet sent is:5

time out,resend

The packet sent is:5

time out,resend

The packet sent is:5

time out,resend

The packet sent is:5

time out,resend

The packet sent is:5

time out,resend

The packet sent is:5

No acknowledgement number:6

The packet sent is:5

Aknowledgement number:6

**CONCLUSION: Through this experiment t=we studied the implementation of ARQ algorithm.**

**ASSIGNMENT NO. 8**

|  |  |
| --- | --- |
| **TITLE:** Study of Networking Components and LAN |  |

**PROBLEM STATEMENT:** Study of Networking Components and LAN

**OBJECTIVE:**

To study various components involved in the communication network.

**THEORY:**

**Introduction:**

* **Computer network** is a group of two or more computers that connect with eachother to share a resource.
* **Sharing of devices and resources is the purpose of computer network.**You canshare printers, fax machines, scanners, network connection, local drives, copiers and other resources.
* In computer network technology, the reare severaltypesofnetworksthatrange

fromsimpletocomplexlevel.

* However,in any case in order to connect computers with each other or to the existing network or planning to insta**ll** from scratch, **the required devices and** **rules (protocols) are mostly the same**.
* Computer network requires the following devices (some of the mare optional):-
  + Repeater
  + Hub
  + Switch
  + Bridge
  + Router
  + Gateway

**Repeater:** A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal.

When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

**Hub:** A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

**Switch:** A switch is a multiport bridge with a buffer and a design that can boost its efficiency (large number of port simply less traffic) and performance. Switch is datalink layer device. Switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

**Bridge:** A bridge operates at data link layer. A bridge is a repeater, with addon functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

**Router:** A router is a device like a switch that routes data packets based on their IPaddresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.

**Gateway:** A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically works as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

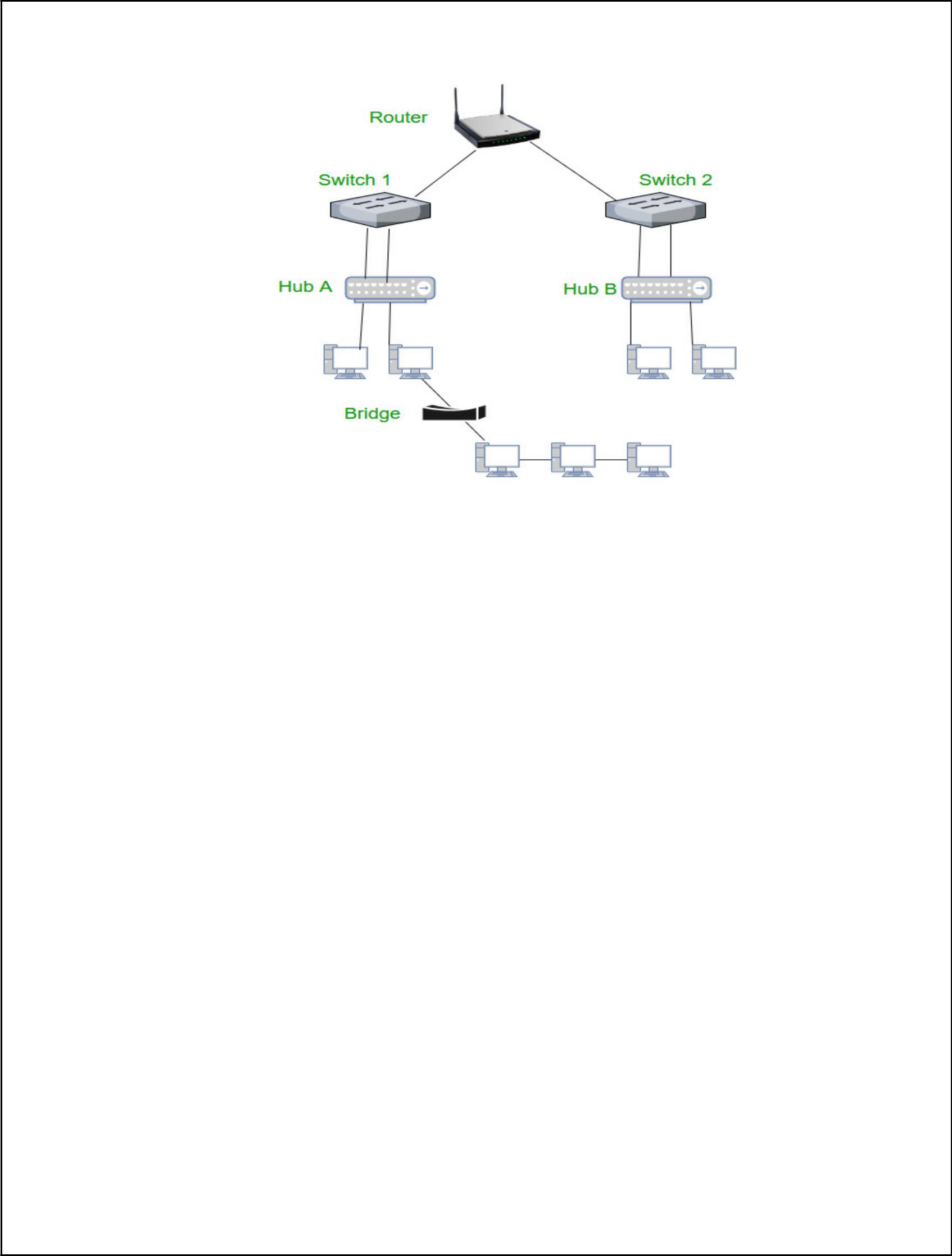
**LAN (Local Area Network)**

* A local area network (LAN) is a devices network that connect with each other in

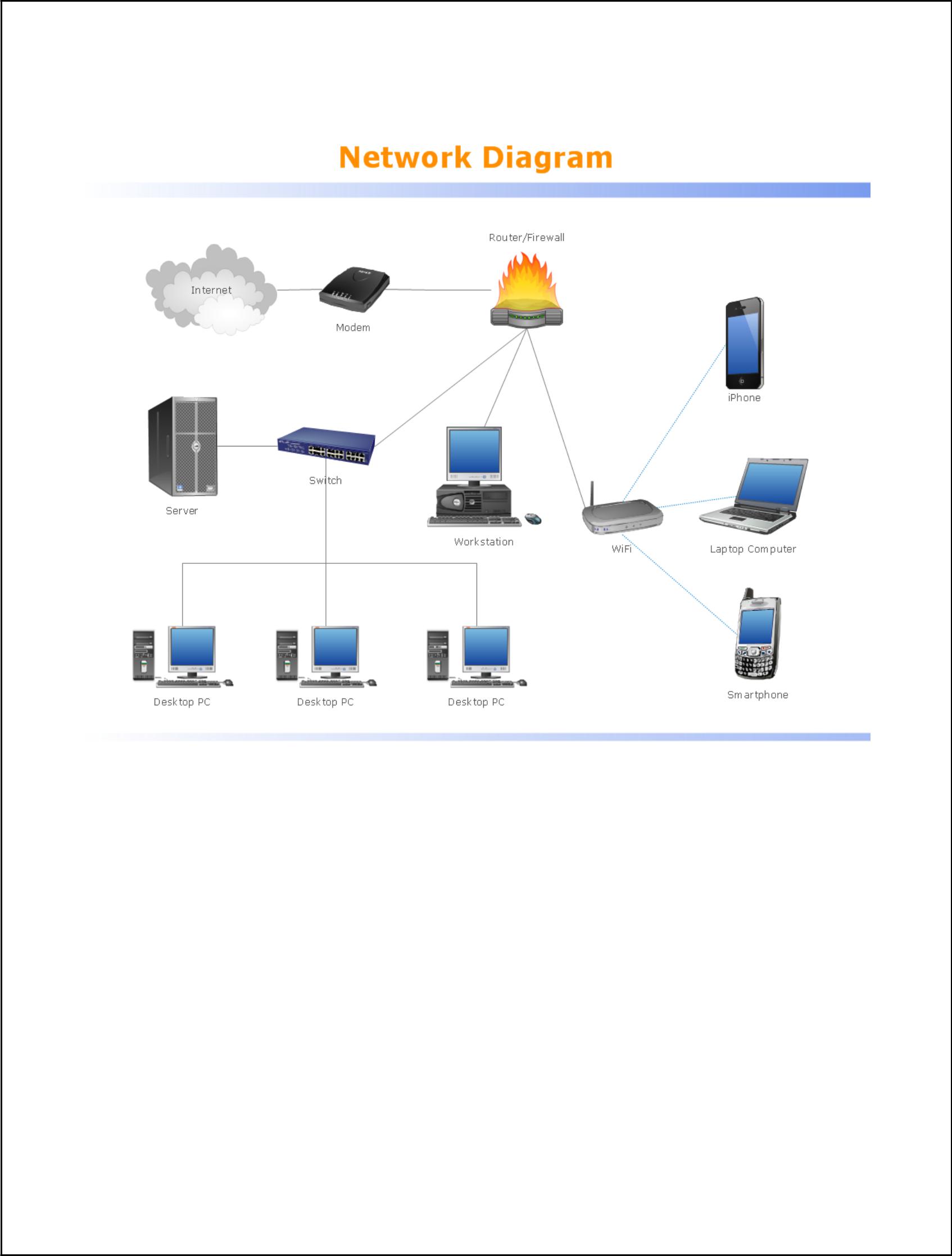
The scope of a home, school, laboratory, or office.

* Usually, a LAN comprise computers and peripheral devices linked to a local domain server. Allnetwork appliances can use a shared printers or disk storage. A local area network serve for many hundreds of users.
* Typically, LAN includes many wires and cables that demand a previously designed network diagram. They are used by IT professionals to visual document the LANs physical structure and arangement.

The Network Logical Structure Diagram is designed to show the logical organization of a network. Shows the basic network components, network structure, and determines the interaction of all network devices. The diagram displays basic devices and zones: Internet, DMZ, LAN, and group. Clarifies what network equipment is connected, describes the major nodes in the network, gives an understanding of the logical structure of the network as well as the type of interaction within the network.

**

***Fig:NetworkComponents***

**

**CONCLUSION: Through this experiment we were able to study various components involved in the communication network.**