Experiment - 1

Aim : ① Write a MATLAB code to find the entropy for given set of probabilities.
② Find the entropy for a binary system and plot the graph of entropy vs probability.

Software : MATLAB

Theory : Entropy is the expected value of the information contained in each message, units of entropy are shannon, n and hartley, depending on the base of log used to define it.

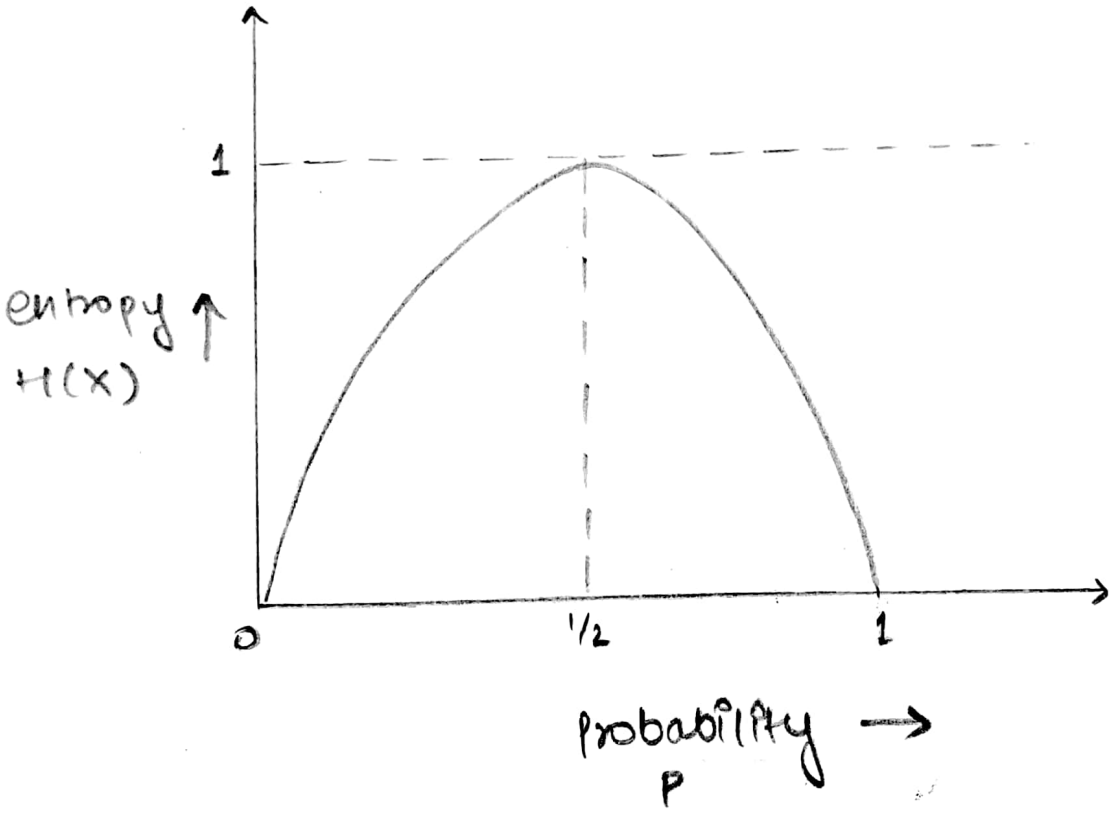
$$H(X) = -\sum_{i=1}^{k} P_i \log(P_i)$$

P_i = probability of i^{th} source
$H(X)$ = entropy

Binary memoryless source

(a) Binary source 0 for probability P_0
(b) Binary source 1 for probability $1-P_0$

$$H(X) = -P_0 \log P_0 - P_1 \log P_1$$



or $H(X) = -P_0 \log P_0 - (1-P_0)\log(1-P_0)$

(1) when probability $P_0 = 0$, entropy $H(X) = 0$

(2) when probability $P_0 = 1$, $H(X) = 0$

entropy contains its maximum value when $H(X) = 1$

or $P_0 = P_1 = \frac{1}{2}$ ie symbols 1 and 0 are equally probable.

Result : Graph between probability and entropy $H(X)$ is made.

09

07+02 = 09

10

Vanita Jain

Experiment No.1

Experiment-2

Aim: write a code in MATLAB for the following:

(a) H(X)
(b) H(Y)
(c) H(X,Y)
(d) H(Y/X)
(e) H(X/Y)

Software: MATLAB

Theory:

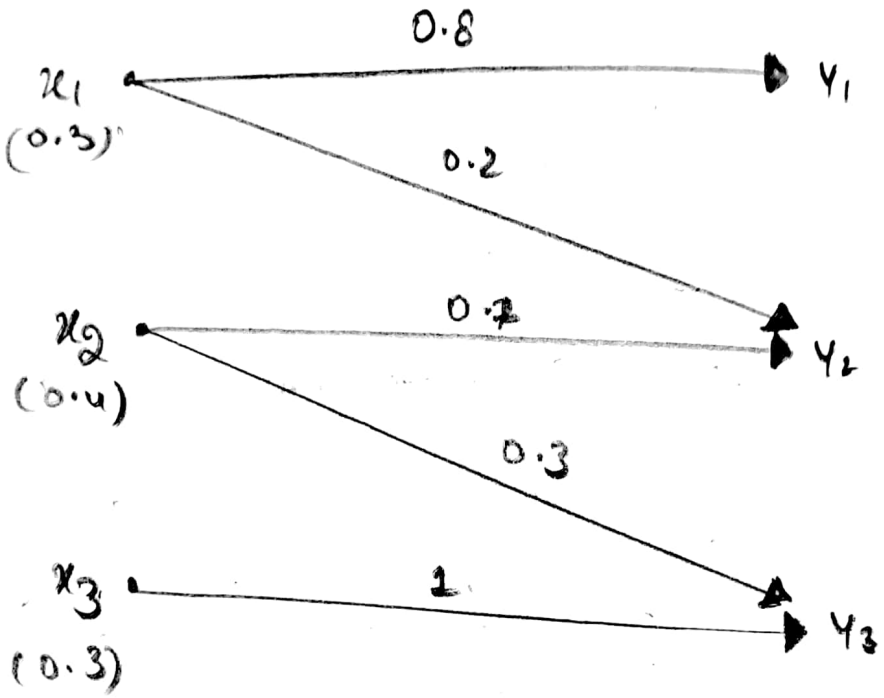
Conditional entropy

→ the average conditional self-information called the conditional entropy is defined as:

$$H(X/Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log P(x_i/y_j)$$

The physical interpretation of the definition is as follows:

H(X/Y) is the information (or uncertainty) in X after Y is observed.



Channel Diagram.

also H(

in

given

H(Y

Joint ent

→ the j

varia

is

H(X,

Result

calc

H(X)

H(Y)

H(X/

H(Y/X

H(X,

Experiment No.1

also $H(Y/X)$ is the information (or uncertainty) in Y after X is observed and is given by:

$$H(Y/X) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_i) \log P(y_i/x_i)$$

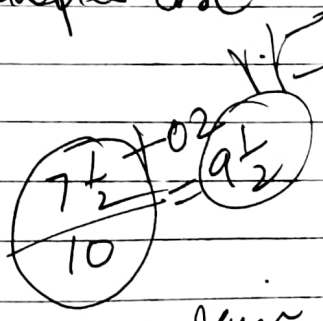
Joint entropy

→ the joint entropy of a pair of discrete random variable (X, Y) with joint distribution $p(x,y)$ is defined as:

$$H(X,Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_i) \log P(x_i, y_i)$$

Result: Joint and conditional entropies are calculated as follows:

$H(X) = 1.5710$ bits/message
$H(Y) = 1.5490$ bits/message
$H(X/Y) = 1.5354$ bits/message
$H(Y/X) = 1.6032$ bits/message
$H(X,Y) = 2.1400$ bits/message



Vanita Jain

Experiment No.1

(a) To find entropy for a binary source.
(b) To find entropy for a given set of probabilities.

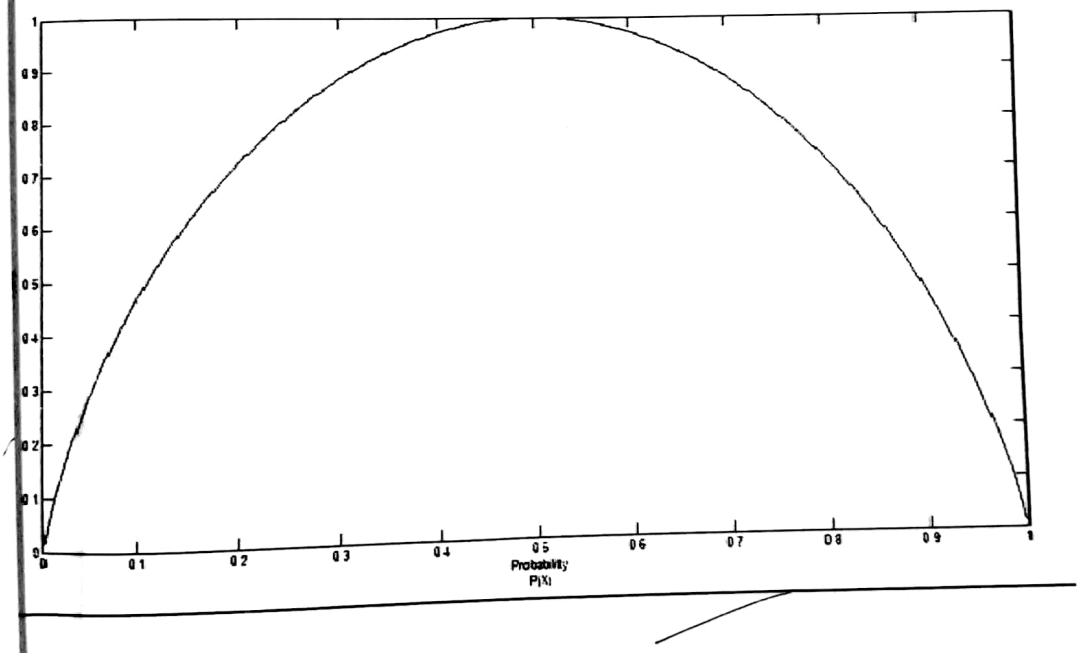
rce Code:

```
 [.001:.001:1];
= 1-p;
 -p.*log2(p)-p2.*log2(p2);
t(p,e);


 [.01 .54 .1 .2 .09 .06];
 entropy(p);
```

5850

ut:



EXPERIMENT – 2

AIM: Write a code in MATLAB for the following entropies:

(a) H(x)
(b) H(y)
(c) H(y/x)
(d) H(x/y)
(e) H(x,y)

CODE:

```
clc;
px= [0.3,0.4,0.3];
disp('p(x)')
disp(px)
hx=sum(-px.*log2(px));
disp('H(x)')
disp(hx)

pybyx = [0.8,0.2,0;0,0.7,0.3;0,0,1];
disp('p(y/x)')
disp(pybyx)
hybyx=sum(NaNsum(-pybyx.*log2(pybyx)));
disp('H(Y/x)')
disp(hybyx)

py=px*pybyx;
disp('p(y)')
disp(py)
hy=sum(NaNsum(-py.*log2(py)));
disp('H(y)')
disp(hy)

pxd=diag(px);
pxy=pxd*pybyx;
disp('p(x,y)')
disp(pxy)
hxy=sum(NaNsum(-pxy.*log2(pxy)));
disp('H(x,y)')
disp(hxy)

pdy = diag(py);
pxbyy = pxy*inv(pdy);
```

Experiment - 3

Aim: To implement Shannon-fano coding

Software used:

Theory:
In the field of data compression, Shannon fano coding is a technique used 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 doesn't achieve the lowest possible expected code word length like huffman coding.

An efficient code can be obtained by following simple procedure:

a) List the source symbols in order of decreasing probabilities
b) Partition the set into two ~~sets~~ that are as close to equiprobable as possible and assign 0 to the upper set and 1 to the lower set.
c) Continue this process each time partitioning the sets with as nearly equal probability as possible until further partitioning is not possible.
d) In shannon fano encoding an ambiguity may arise in choice of approximately equiprobable sets.

Implementation :

Probability				
0.4	0			0
0.3	1	0		10
0.2	1	1	0	110
0.1	1	1	1	111

Entropy

$$H(X) = -\sum_{i=1}^{n} P_i \log P_i$$

$$= 1.8464 \text{ bits / message}$$

$$\bar{L} = \sum_{i=1}^{n} n_i P_i$$

$$= 1.900$$

$$\eta = H(x) / \bar{L}$$

$$= .9718$$

Result : Successfully implemented shannon-fano coding in MATLAB.

09/10

14/8/17

EXPERIMENT-3

AIM

Write a program in MATLAB to perform Shannon Fano Coding

MATLAB Code

```
function code2 = createCode(p)
    start_index = 1;
    end_index = length(p);
    total_length = length(p);
    code = {};
    i = 1;
    while(i<=total_length)
        code = horzcat(code, {''});
        i = i + 1;
    end
    code2 = findBreakingIndex(p, code, start_index, end_index);
end
-----------------------------------------------------------------
function code = findBreakingIndex(p,code, startIndex, endIndex)
    if (endIndex - startIndex == 0)
        return
    end

    originalStartIndex = startIndex;
    originalEndIndex = endIndex;
    sum1 = p(startIndex);
    sum2 = p(endIndex);
    i=0;
    l = (endIndex-startIndex);
    while(i<=l)
        if(sum1 > sum2)
            code(endIndex) = strcat(code(endIndex), '1');
            endIndex = endIndex - 1;
            sum2 = sum2 + p(endIndex);
        else
            code(startIndex) = strcat(code(startIndex), '0');
            startIndex = startIndex + 1;
            sum1 = sum1 + p(startIndex);
        end
        i = i +1 ;
    end
    startIndex = startIndex-1;
    endIndex  = endIndex+1;
    index = startIndex;
    code = findBreakingIndex(p, code, originalStartIndex, index);
    code = findBreakingIndex(p, code, endIndex, originalEndIndex);
-----------------------------------------------------------------

function h = findEntropy(p)
    i = 1;
    len = length(p);
    sum = 0;
    while(i<=len)
        sum = sum + p(i)*log2(p(i));
        i = i + 1;
```

```
    end
    h = -sum;
end

function avgLength = findAvgLength(code, p)
    i = 1;
    len = length(code);
    sum = 0;
    while(i<=len)
        sum = sum + length(char(code(i)))*p(i);
        i = i + 1;
    end
    avgLength = sum;
end

function n = findEfficiency(code, p)
    avgLength = findAvgLength(code, p);
    h = findEntropy(p);
    n = h / avgLength;
end
```

OUTPUT:

```
p=[0.4 0.3 0.2 0.1]

c = createCode(p);
>> c
c =

    '0'    '10'    '110'    '111'
>> hx = findEntropy(p);
>> hx
hx =

    1.8464
>> l = findAvgLength(c,p);
>> l
l =

    1.9000
>> n = findEfficiency(c,p);
>> n
n=
    0.9718
```

Mrinalini
03111503113

Experiment 4

Date :
Page. No.

Experiment - 4

Aim: Write a MATLAB code to insert parity bits in a given message for hamming code.

Software: MATLAB 7.10.0

Theory:

Hamming codes are a family of linear error correcting codes that generalise the hamming (7,4) - code. It can detect upto two bit errors or correct one-bit errors without detection of uncorrected errors. By contrast the simple parity code cannot correct errors and can detect only an odd no. of bits in errors. Hamming codes are perfect codes and achieve highest possible rate for codes with their block length and minimum distance of three.

Hamming code is a set of error correcting codes that can be used to detect and correct bit errors that can occur when computer data is moved or stored. Like other error correcting codes, hamming codes make use of parity bits. Using more than one parity bit an error correction code cannot only identify a single bit error in the data unit but also its location.

Experiment 4

Date :
Page. No.

Hamming codes are a class of binary linear codes. For each integer $r \geq 2$ there is a code with a block length $n = 2^r - 1$ and message length $k = 2^r - r - 1$. Hence the rate of hamming codes is $R = \frac{k}{n} = \frac{(1-r)}{2^r - 1}$ which is highest possible for codes with minimum distance of 3 and block length $2^r - 1$. The parity check matrix of a hamming code is constructed by listing all columns of length r that are non-zero and this matrix has the property that any two columns are pairwise linear independent.

Result : Hamming codes are generated successfully.

Experiment 4

Aim: To implement Hamming code using MATLAB

Code:

```
clc
m=input('Enter the message bits:')
p=input('Enter the no. of parity bits:')

mn=length(m)
n= ((2^p)-1)
y=zeros(1,n);
for i=1:p
 y(2^(i-1))=NaN;
end
a=1;
for j=1:(mn+p)
   if(isnan(y(j)))
    continue;
   else
    y(j)=m(a);
    a=a+1;
  end
end
y
i=1;
pc=zeros(1,p)
for k=1:n
 if(y(k)==1)
   pc(i)=k;
   i=i+1;
end
end
pc1=length(pc);
sf=dec2bin(pc,p);
sf';
k=sum(sf)

h=zeros(1,p);
for i=1:p
 if(rem(k(i),2)==0)
h(i)=0;
else h(i)=1;
end
end
h
a=1;
for j=1:(mn+p)
if(isnan(y(j)))
y(j)=h(a);
a=a+1;
end
end
y
```

Output:

```
Enter the message bits:[1 0 1 1 0 0 1]
m =
     1     0     1     1     0     0     1
Enter the no. of parity bits:4
p =
     4
mn =
     7
n =
    15
y =
   NaN   NaN     1   NaN     0     1     1   NaN     0     0     1     0     0     0     0
pc =
     0     0     0     0
k =
   193   194   196   195
h =
     1     0     0     1
y =
     1     0     1     0     0     1     1     1     0     0     1     0     0     0     0
```

Date :
Page. No.

Experiment - 6

Aim : Write a code in MATLAB to generate linear block code.

Software : MATLAB 7.10.0

Theory :

A linear code is an error correcting code for which any linear combination of code-words is also a codeword.

Linear codes are traditionally partitioned into block codes and convolution codes, although turbo codes can be seen as a hybrid of these two types.

Linear codes are used in forward error correction and are applied in methods for transmitting symbols on a communication channel.

Functions used :

(1) rem : remainder after division
Syntax : r = rem(a, b)

Date :
Page. No.

It returns the remainder after division of 'a' by 'b' where a is the dividend and b is the divisor.

Result :
Linear block codes generated in MATLAB.

09
10

Vj

Experiment 5

Aim: Write a code in MATLAB to generate linear block code.

MATLAB code:

```
clc;
g=input('enter the g matrix');
disp('g=');
disp(g);
a=[0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15];
b=dec2bin(a)
c=rem(b*g,2)
disp('min hamming distance: ');
d_min=min(sum((c(2:2^k,:))'));
disp(d_min)
```

OUTPUT

enter the g matrix[1 1 0 1 0 0 0;0 1 1 0 1 0 0;0 0 1 1 0 1 0;0 0 0 1 1 0 1]

g=

```
   1   1   0   1   0   0   0
   0   1   1   0   1   0   0
   0   0   1   1   0   1   0
   0   0   0   1   1   0   1
```

b =

```
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

c =

0	0	0	0	0	0	0
0	0	0	1	1	0	1
0	0	1	1	0	1	0
0	0	1	0	1	1	1
0	1	1	0	1	0	0
0	1	1	1	0	0	1
0	1	0	1	1	1	0
0	1	0	0	0	1	1
1	1	0	1	0	0	0
1	1	0	0	1	0	1
1	1	1	0	0	1	0
1	1	1	1	1	1	1
1	0	1	1	1	0	0
1	0	1	0	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	1

min hamming distance:

3

Date :
Page. No.

Experiment - 6

Aim : Write a code in MATLAB to implement Viterbi algorithm for decoding a message.

Software used : MATLAB 7.10.0

Theory :

Viterbi algorithm is an efficient way to decode convolution codes. It avoids the explicit enumeration of 2^N possible combinations of N-bit parity bit sequences.

Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states called Viterbi path that results in the context of Markov information sources and hidden markov models.

poly2trellis()
It converts convolution code polynomials to trellis description.

Date :
Page. No.

Syntax

trellis = poly2trellis (constraint length, code generator)

vitdec ()

It convolutionally decodes binary data using viterbi algorithm.

Syntax

decoded = vitdec (code, t, th, 'trunc', 'hard')

Result :

Viterbi algorithm was implemented successfully

9½/10

Experiment No.6

Aim: Write a code in MATLAB to implement viterbi algorithm for decoding a message.

Source Code:

```
clc;
t=poly2trellis([4 3],[4 5 17;7 4 2]);
code = convenc([1 0 1 0 1 1],t);
tb = 2;
code
decode = vitdec(code,t,tb,'trunc','hard')
```

Output:

```
code =

  0   0   1   1   1   0   0   0   1

decode =

  1   0   1   0   1   1
```

Date :
Page. No.

Experiment - 7

Aim: Write a MATLAB code to generate Binary cyclic code.

Software : MATLAB 7.10.0

Theory: Cyclic codes form an important subclass of linear codes. These codes are attractive for 2 reasons: encoding and syndrome computation can be implemented easily by implementing shift register with feedback connections, and second, because they have considerable inherent algebraic structure, it is possible to find various practical methods for decoding them.

Cyclic codes are special linear codes of large interest and importance because

- they possess a rich algebraic structure that can be utilised in a variety of ways.
- They have extremely concise specification.
- They can be efficiently implemented using simple shift registers
- most of the practically important codes are cyclic.

Date :
Page. No.

channel codes allow to encode streams of data
In order to specify a binary code with 2^k codewords of length n one may need to write down 2^k code words of length n.

In order to specify a linear binary code of the dimension k with 2^k codewords of length n, it is sufficient to write down k codewords of length n.

In order to specify a binary cyclic code with 2^k codewords of length n, it is sufficient to write 1 codeword of length n.

A code C is cyclic if
- C is a linear code
- any cyclic shift of a codeword is also a codeword.

eg. $C = \{000, 011, 110, 101\}$

Result: cyclic codes are generated successfully.

10/10 VJ

Experiment No.9

Aim: Write a MATLAB code to generate Binary cyclic code.

Source Code:

```
clc;
n=input('Enter the codeword length:');
k = input('Enter the no of message bits:');
pol = cyclpoly(n,k);

disp('--systematic binary cyclic code.')
[parmat,genmat,k] = cyclgen(n,pol)

disp('--Nonsystematic binary cyclic code.')
[parmatnonsys,genmatnonsys,k] = cyclgen(7,pol,'nonsys')
```

Output:

```
Enter the codeword length:7
Enter the no of message bits:4
--systematic binary cyclic code.
parmat =

   1   0   0   1   1   1   0
   0   1   0   0   1   1   1
   0   0   1   1   1   0   1

genmat =

   1   0   1   1   0   0   0
   1   1   1   0   1   0   0
   1   1   0   0   0   1   0
   0   1   1   0   0   0   1

k = 4
--Nonsystematic binary cyclic code.
parmatnonsys =

   1   1   1   0   1   0   0
   0   1   1   1   0   1   0
   0   0   1   1   1   0   1

genmatnonsys =

   1   0   1   1   0   0   0
   0   1   0   1   1   0   0
   0   0   1   0   1   1   0
   0   0   0   1   0   1   1

k = 4
```

Experiment - 8

Aim : Write a code in MATLAB to generate BCH

Software : MATLAB 7.10.0

Theory : In coding theory the BCH codes form a class of error correcting codes that are constructed using finite fields. One of the key feature in BCH codes is that during code design there is a precise control over the number of symbol errors correctable by the code. In particular it is possible to design binary BCH codes that is the case with which they can be decoded namely via an algebraic method known as syndrome decoding. This simplifies the design of the decoder for these codes, using small low power electronic hardware.

BCH codes are used in applications such as satellite communications, compact disk players, DVDs, disk drives, solid state drives and two dimensional bar codes.

Given a prime power 'q' and positive integers m and d with $d \leq q^m - 1$, a primitive narrow-

Bunty

Experiment 8

Aim: Write a code in MATLAB to generate BCH.

MATLAB code:

```
clc;
x=input('enter the sequence: ');
g=gf(x);
n= input('enter the no. of bits:');
k=4;
code=bchenc(g,n,k)
msg=bchdec(code,n,k)
```

OUTPUT

enter the sequence:[1 1 1 0]
enter the no. of bits:7

code = GF(2) array.

Array elements =

1 1 1 0 1 0 0

code = GF(2) array.

Array elements =

1 1 1 0 1 0 0

Mrinalini
03111503113

sense BCH code over the finite field (or Galois field) GF(q) with code length $n = q^m - 1$ & distance at least d is constructed

For any integer $m \geq 3$ & $t \leq 2^{m-1}$ there exists a positive BCH code with following parameters:

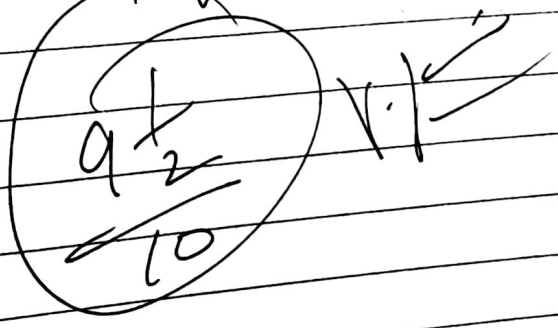
$$n = 2^m - 1$$

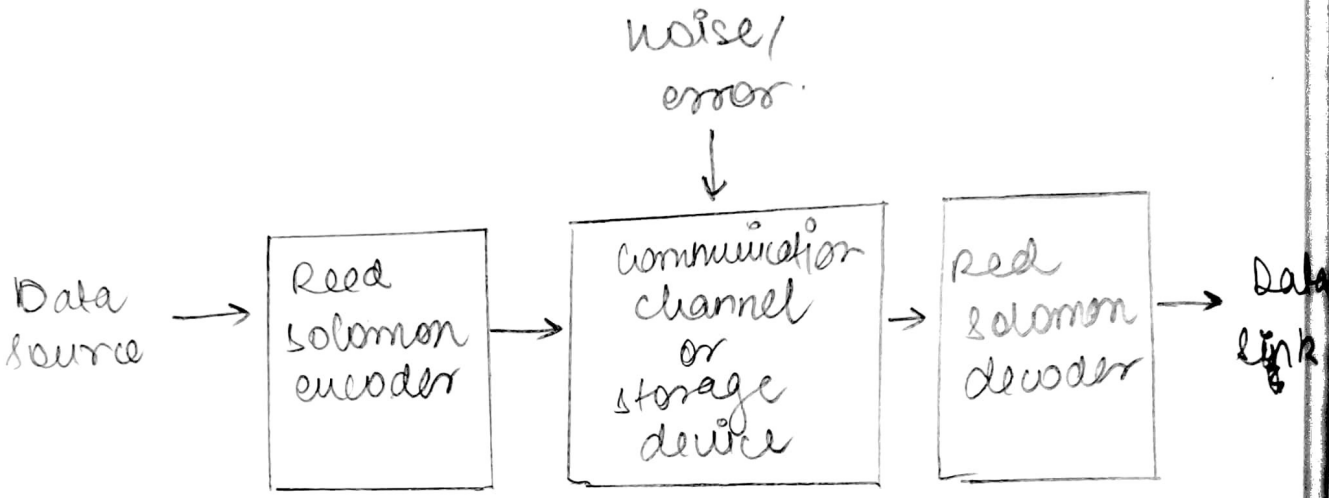
$$n - k \leq mt$$

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

This code can correct it or fewer random errors over a span of $2^m - 1$ bit positions. This code is a t-error-correcting BCH code.

Result: BCH code is performed successfully on MATLAB.





solomon encode - decoder system.

Experiment - 9

Aim: Write a code in MATLAB to generate Reed Solomon code.

Software: MATLAB 7.10.0

Theory: Reed solomon codes are block based error correcting codes with a high or wide range of applications in digital communications and storage. These codes are used to correct errors in many systems including:

- Storage devices (DVD, CD, tape etc)
- Wireless or mobile communications (telephones, microwave links)
- Satellite communications.
- High speed modems such as ADSL, xDSL etc.

The Reed solomon encoder takes the block of digital data and add extra redundant bits. Error occured during transmission or storage for a no. of reasons. The Reed solomon decoder processes each block & attempts to correct errors & recover the original data. The no. & the types of errors that can be corrected depends on characteristics

of reed solomon code.

these code belong to the class of non-binary cyclic error-correcting codes. The Reed-Solomon code is based univariable polynomials over finite fields.

It is able to detect correct multiple symbol errors. By adding t check symbols to the data, it can detect any combination of up to 't' erroneous symbols or correct upto $\lfloor t/2 \rfloor$ symbols.

gf : It is used to create a Galois field array
Syntax ⇒ msg = gf(x, m) where it converts x field array from matrix 'x' with 2^m elements if $1 < m \leq 16$ elements of x should be integers b/w 0 to $2^m - 1$.

rsenc : Reed solomon encoder fn
rsenc (msg, n, k)

rsdec : Reed solomon decoder fn
rsdec (code, n, k)

Result : Reed solomon code is studied & implemented successfully.

GOOD WRITE

Experiment 9

Aim: Write a code in MATLAB to generate the Reed Solomon code.

MATLAB code:

```
clc;
n=7;
k=3;
m=3;
msg=gf([1 2 3;5 5 6],m);
c=rsenc(msg,n,k)
code=rsdec(c,n,k)
```

OUTPUT

c = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

1	2	2	3	1	3	0
4	5	6	6	7	5	7

code = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

1	2	2
4	5	6

Mrinalini
03111503113