SHREE RAMCHANDRA EDUCATION SOCIETY'S

SHREE RAMCHANDRA COLLEGE OF ENGINEERING, LONIKAND, PUNE – 412 216

DEPARTMENY OF ELECTRONICS AND TELECOMMUNICATION

LAB MANUAL

Information Theory & Coding Techniques (Semester-II)

SHREE RAMCHANDRA COLLEGE OF ENGINEERING, PUNE

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

…….....

CLASS:-TE (E&TC) **SUB: - ITCT.**

LIST OF PRACTICALS

1. 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 – Fanocodingand decoding
   b) Huffman Coding and decoding
   c) Lempel Ziv Coding and decoding
3. Write a Program for coding & decoding of Linear block codes.
4. Write a Program for coding & decoding of Cyclic codes.
5. Write a program for coding and decoding of convolutional codes.
6. Write a program for coding and decoding of BCH and RS codes.
7. Write a program to study performance of a coded and uncoded communication system (Calculate the error probability).
8. Write a simulation program to implement source coding and channel coding for transmitting a text file.

SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 01

Name:	**Date of performance:**
Class:	**Date of Submission:**
Roll no:	**Signature:**

Title: 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.

.

Software Requirement: MATAB/C

THEORY:

Information – The probability denotes likeliness or the certainty of occurrence of any event. A less probable event is rarer and so it contains more information. Thus, if n event of lower probability occurs, it conveys more information than occurrence of an event of larger probability.

If 'p' is the probability of occurrence of the message symbol and '1' is the information received from the message, then;

Information , $\mathbf{I = \log_2(1/p)}$

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

Entropy (Average information) : Suppose there are m different msg m1, m2, m3, …….m_m having probabilities p1, p2,p3, …..pm. suppose a sequence of L message is transmitted. If L is very large then we can say that msg of m are transmitted.

$I1= p1[\log_2 (1/p1)]$

$I2= p2[\log_2 (1/p2)]$

…..

……

……

$Im= pm[\log_2 (1/pm)]$

Thus the total information is

Itotal = I1+I2+…….+ Im

Entropy (H) = I_L (total)

Where I_L (total)= total information of no. of messages.

Information Rate: It is given by

R= r*H bits/sec

Where R= Rate at which msgs are generated

H = Entropy

Conclusion:

Program:

Program –

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int a,b,i,j;
float p[15][15]={0}, x[15]={0}, q[15][15]={0}, y[15]={0};
clrscr();
printf("\nEnter number of input and output ");
scanf("%d%d",&i,&j);
printf("\nEnter the elements of joint probability matrix row wise: ");
for(a=0; a<i;a++)
{
for(b=0; b<j;b++)
{
printf("\np[%d][%d]=",a,b);
scanf("\t%f",&p[a][b]);
}
}
printf("\n\np[x,y]");
for(a=0;a<i;a++)
{
printf("\n");
for(b=0; b<j;b++)
{
printf("\t%.4f",p[a][b]);
}
}
printf("\n\nThe probability of inputs: ");
for(a=0; a<i;a++)
{
printf("\n p(x[%d]) :",a);
for(b=0;b<j;b++)
x[a]=x[a]+p[a][b];
}
printf("\n\nThe probability of outputs: ");
for(b=0; b<j;b++)
{
printf("\n p(y[%d]) :",b);
for(a=0;a<i;a++)
y[b]=y[b]+p[a][b];
printf("%0.4f",y[b]);
}
printf("\n\nConditional probability matrix p(y/x) :");
for(a=0;a<i;a++)
{ printf("\n");
for(b=0;b<j;b++)
{
```

```
q[a][b]=p[a][b]/x[a];
printf("\t%0.4f",q[a][b]);
}
}
getch();
}
```

// Write a Program to implement an algorithm for Determination of Entropy, Information, Information Rate.

Program –

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int i,n,r;
float I[10],p[10],H[10],s=0,TI,TH;
clrscr();
printf("\n\n Enter number of symbols:");
scanf("%d",&n);
xx:printf("\n\n Enter the probablities:-\n");
for(i=0;i<n;i++)
scanf("%f",&p[i]);
for(i=0;i<n;i++)
s=s+p[i];
printf("\n\n Sum of probabilities : %f",s);
if(s==1)
{
for(i=0,TI=0;i<n;i++)
{
I[i]=log(1/p[i])/log(2);
printf("\nI[%d]=%f",i,I[i]);
TI+=I[i];
}
for(i=0,TH=0;i<n;i++)
{
H[i]=p[i]*(log(1/p[i])/log(2));
printf("\nH[%d]=%f",i,H[i]);
TH+=H[i];
}
printf("\nEntropy : %0.3f bits/sample",TH);
printf("\nMutal Imformation : %0.3f bits/sample",TH);
printf("\n\n Enter the rate r (samples/sec):");
scanf("%d",&r);
printf("\n\nInformation rate is %f bits/sec",r*TH);
}
else
{
printf("\n\nEntered probablities are not correct please re-enter");
goto xx;
}
getch();
```

SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 02

Name: **Date of performance:**

Class: **Date of Submission:**

Roll no: **Signature:**

1. **Title:** Write a program for generation and evaluation of variable length source coding using
   a) Shannon – Fanocoding and decoding
   b) Huffman Coding and decoding

.**Objective:**

1. Calculate the efficiency of Shanon Fano Coding.
2. Calculate the efficiency of Huffman Coding.

Software Requirement: MATAB/C

THEORY:

SHANNON- FANO CODING:

It is a type of source coding Shannon theorem. The design of variable length code such that its average code word length approaches the entropy of discrete less source is called entropy coding.

An efficient code can be obtained by this method:

1. List the source symbol in order of decreasing probability.
2. Partition the set into two set that are as close as equiprobable and assign 0 to the each msg. in upper set and 1 to each msg. in lower set.
3. Continue this process each time partitioning the sets with nearly equal probabilities as nearly equal time partitioning is possible.

Algorithm:

1. Input ‘n’ number of messages.
2. Input the values of probabilities.
3. Arrange the messages in decreasing order of probabilities.
4. Divide the mes. Into two equiprobable set X1 and X2
5. Assign ‘0’ to all msg. in X1.
6. Assign ‘1’ to all msg. in X2.
7. Repeat till single msh. Is left out.
8. Print the code of msg.
9. Calculate H(x), L and n
10. Stop .

HUFFMAN CODING:

Huffman coding is one of the efficient coding techniques which is variable length coding is to assign each symbol of an alphabet sequence of bits roughly equal in length to the amount of information conveyed by the symbol ‘n’ question. The end result is a source code whose avg. code word length approaches the entropy H(x) of that source.

Algorithm:

1. Start.
2. Input the total number of probabilities.
3. Arrange the messages in decreasing order of probabilities.
4. Add last two probabilities.
5. Assign them ‘0’ and ‘1’.
6. With addition & other probabilities again sort out the total probabilities.
7. If the addition result is equal to probability of an symbol then put it on the top
8. Repeat the program from step 4 until addition is 1.
9. To find code for particular symbol take the path of probability of symbol and write cod in reverse fashion.
10. Find out entropy, avg. code word length and efficiency.
11. Stop .

Conclusion:

\

Program:

```
// Shannon fano coding
#include<stdio.h>
#include<conio.h>
#include<math.h>
void q(float c[20][20],int low,int high,int k,float p[20]);
void main()
{
int m,p1,low=1,k=0,high,r,s,i,n,count1[20]={0};
float c[20][20]={5},p[20],temp,h=0,l=0,efficiency;
clrscr();
printf("\nenter the no. of messages : ");
scanf("%d",&high);
for(m=0;m<high;m++)
{
for(p1=0;p1<10;p1++)
c[m][p1]=5;
}
for(m=1;m<(high+1);m++)
{
printf("\nenter the prob of p[%d] : ",m);
scanf("%f",&p[m]);
}
for(r=1;r<high+1;r++)
{
for(s=1;s<high+1-r;s++)
{
if(p[s+1]>p[s])
{
temp=p[s+1];
p[s+1]=p[s];
p[s]=temp;
}
```

```
}
}
for(i=1;i<high+1;i++)
c[i][0]=p[i];
q(c,low,high,k,p);
printf("\n");
printf("\nprobability\tsource code");
for(m=0;m<high+1;m++)
{
for(p1=0;p1<6;p1++)
{
if(c[m][p1]==5)
printf(" ");
else
{
if((c[m][p1]==0)||(c[m][p1]==1))
{
count1[m-1]++;
printf("%.0f\t",c[m][p1]);
}
else
printf("%.5f\t--->",c[m][p1]);
}
}
printf("\n\n");
}
for(i=1;i<high+1;i++)
{
h+=p[i]*(log(1/p[i])/log(2));
l+=p[i]*count1[i-1];
}
efficiency=(h/l)*100;
printf("\n\nh=%.4f\n\nl=%.4f\n\nefficiency=%.2f",h,l,efficiency);
printf("%");
```

```
getch();
}
void q(float c[20][20],int low,int high,int k,float p[20])
{
float s1=0,s2=0,s3=0;
int count,count1,p1,m,i,j,first,last;
if(low<high)
{
k++;
first=low;
last=high;
count=first-1;
for(i=first;i<(last+1);i++)
s1=s1+p[i];
s2=s1/2;
for(j=first;j<(last+1);j++)
{
s3=s3+p[j];
if(s3<=s2)
{
c[j][k]=0;
count++;
}
else
c[j][k]=1;
}
q(c,first,count,k,p);
q(c,count+1,high,k,p);
}
}
```

```
%  HUFFMAN CODING

clc;
clear all;
close all;
code_length=0;
x=input('Enter number of symbols: ');
for m=1:x
   symbols(m)=input('Enter the symbol number: ');
   p(m)=input('Enter the probability: ');
end
Hx=0;
for m=1:x
   [dict,avglen]=huffmandict(symbols,p);
   hcode=huffmanenco(m,dict)
   dsig = huffmandeco(hcode,dict)
   code_length=length(hcode)
   Hx=Hx+(p(m)*(-log(p(m)))/(log(2)));
end
display(Hx);
Efficiency=(Hx/avglen)*100
```

SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 03

Name:	**Date of performance:**
Class:	**Date of Submission:**
Roll no:	**Signature:**

Title: Write a Program for coding & decoding of Linear block codes.

.**Objective:**

Error detecting and correcting using liner block code.

Software Requirement: MATAB/C

THEORY:

Linear Block Code:

A code is a linear if the sum c mod 2 addition of any two code vectors produces another code vector i.e. any code vector can be expressed as linear combination of other code vector . consider for a (7,4) linear block codes , a particular code vector consists of M1, M2……Mk msg bits and C1, C2, ……..Cq check bits. Then this code vector can be represented as,

X = (M1,M2,M3…….Mk C1,C2,C3……Cq)

Here q= n-k i.e.no.of redundant bits added by encoder. The code vector can be also written as,

X= (MC)

Here M= k- msg. bits

C = q bits check vector

Linear block code may be systematic or non-systematic. In any case

X =MG

Here X= code vector of 1xn size

M= msg. vector of 1x k size

G= generator matrix with size k x n.

Algorithm:

1. Start
2. Accept size of LBC block code in terms n and k
3. Accept parity p matrix of size k x (n-k)
4. Generate generator matrix such that G = [Ik | P] Of size k x n in which Ik is an identity matrix.
5. Generate parity check matrix such that H= [P^T | In-k] Of size (n-k) x n in which P^T is an transpose of P matrix.
6. Generate msg. vector
7. Generate code vector by formula, C = MG
8. Display it
9. Also calculate hamming weight of each code word and that is done by calculating total no. of ones in the code vector. Display it.
10. Calculate detecting capability by Td =dmin- 1 , where dmin is minimum hamming distance.
11. Calculate error correcting capability tc by, tc= (dmin-1) /2
12. Display parity matrix H
13. Calculate syndrome vector for different error pattern 'E'. S= $E.H^T$
14. Compare this S with each S created for different error pattern where S is matched error is in that respective bit w.r.t. the error pattern. Display no. of bits where is there. If S of received vector is '0' then display received vector is correct.

Conclusion:

Program:

```
% LINEAR BLOCK CODE

clc;
clear all;
% input generator matrix
g=input('enter the generatior matrix=')
disp('the order of LBC for given generator matrix=');
[n,k]=size(transpose(g))
for i=1:2^k
   for j=k:-1:1
      if rem(i-1,2^(-j+k+1))>=2^(-j+k)
         u(i,j)=1;
      else
         u(i,j)=0;
      end;
   end;
end;
u
disp('the possible codeword are')
c=rem(u*g,2)
disp('the minimum hamming distance dmin for given block code is=')
d_min=min(sum(c(2:2^k,:)))
%cw
r=input('enter the received codeword=')
p=[g(:,n-k+1:n)]
h=[transpose(p),eye(n-k)];
disp('hamming code=')
ht=transpose(h)
disp('syndrome of given codeword is=');
s=rem(r*ht,2)
for i=1:1:size(ht)
   if(ht(i,1:3)==s)
      r(i)=1-r(i);
      break;
   end;
end;
disp('the error is in bit:'I')
disp('the corrected cw is:' r')
```

SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 04

Name:	**Date of performance:**
Class:	**Date of Submission:**
Roll no:	**Signature:**

Title: Write a Program for coding & decoding of Cyclic codes.

Objective:

Error detecting and correcting using Cyclic code.

Software Requirement: MATAB/C

THEORY:

Cyclic Code:

Cyclic code are the sub-class of linear block codes. They have a property that a cyclic shift of one code word produces another code word. Suppose there is an n-bit code vector.

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

Here **$x_{n-1}, x_{n-2}, \ldots\ldots\ldots\ldots\ldots\ldots x_1, x_0$** represent the individual bits of the code vector X. if the code vector is shifted cyclically , then another code vector X is obtained

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

Algorithm:

1. Start
2. Get the values of n & k.
3. Get the generator polynomial i.e. its coefficient from user.
4. Get the message vector.
5. Get the message generator matrix.
6. Multiply message polynomial with msg bit shifted by n-k
7. Divide this term i.e. X^{n-k} d(x) by g(x)
8. To get code word polynomial add X^{n-k} d(x) with reminder of division.
9. Display the code word.
10. Generate the error pattern & corresponding syndrome with displaying.

11. Enter the received code vector.
12. Divide the received code vector polynomial by generator polynomial.
13. The reminder of division will be the syndrome polynomial.
14. From syndrome detect the corresponding error pattern.
15. Stop.

Conclusion:

Program:

```
% Encoding & Decoding for (7,4) Cyclic code
clc;
clear all;
%Encoding
n=7;
k=4;
p=[1 1 0 ; 1 1 1; 0 0 1 ; 1 0 1]; % Parity Matix
d=[1 1 0 1]; % Message word
ik=eye(k);
g=cat(2,ik,p);disp('Generator Matrix:');disp(g);
g1=cyclpoly(n,k,'max');
gp=poly2sym(g1);
disp('Generator Polynomial:');disp(gp);
c1=mtimes(d,g);
c=mod(c1,2);
disp('The codeword for given message is:'); disp(c);
%Decoding
r=[1 0 0 1 1 1 0];disp('received word of 7 bit:');disp(r);
ink=eye(n-k);
h=cat(2,p',ink);
ht=h';disp('Transpose of parity check matrics :');disp(ht);
rp=poly2sym(r);
[qp,remp]=quorem(rp,gp);
disp('Syndrome polynomial:');disp(qp);
rem=sym2poly(remp);
s=mod(rem,2);disp('Syndrome :');disp(s);
if (s == 0)
disp('The received code is correct.');
else
disp('The received code is incorrect.');
row = 0;
for j=1:1:n
m=xor(s,ht(j,:));
if (m==0)
row = j;
break;
end
end
r(1,row) = ~r(1,row);
disp(r);
disp('Correct codeword is:');

end
disp('c=');
disp(r);
```

SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 05

Name: **Date of performance:**

Class: **Date of Submission:**

Roll no: **Signature:**

Title: Write a program for coding and decoding of convolutional codes.

Objective:

Error detecting and correcting using convolutional code.

Software Requirement: MATAB/C

THEORY:

Convolution Code:

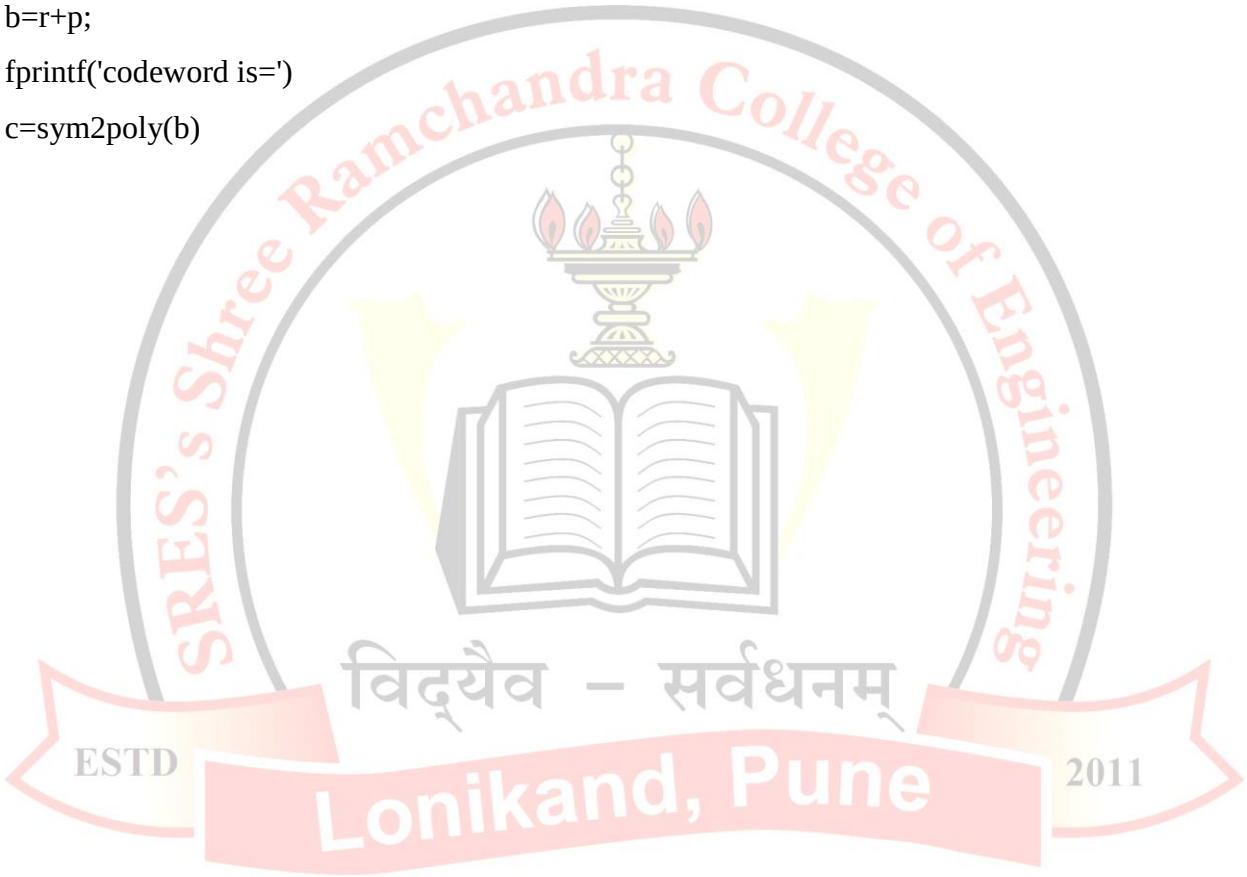
Convolution coding is an alternative to block codes. They differ from block codes in that the encoder contains memory. It means encoder output at any given time is dependent on presentas well as past inputs.

Convolution codes are commonly specified by three parameter (n , k, m) where n is no. of outputs bits(coded), k is no. of inputs bits(msg.), m is memory order.

Program:

```
%cyclic convolution
clc;
clear all;
k=input('ennter the no. of message bits k=');
q=input('given data q=');
fprintf('data polynomial is')
d=poly2sym(q)
n=input('enter the no. of information bits n=');
w=[1,0,0,0];
```

```
x=poly2sym(w);
e=cyclpoly(n,k);
fprintf('generator polynomial ')
g=poly2sym(e)
z=conv(w,q);
r=poly2sym(z)
[m,v]=gfdeconv(z,e);
fprintf('polynomial')
p=poly2sym(v)
b=r+p;
fprintf('codeword is=')
c=sym2poly(b)
```



SRES's

Shree Ramchandra College of Engineering,

Lonikand, Pune

Experiment No. 06

Name: **Date of performance:**

Class: **Date of Submission:**

Roll no: **Signature:**

Title: Write a program for coding and decoding of BCH and RS codes.

Objective:

Error detecting and correcting using BCH and RS codes code.

Software Requirement: MATAB/C

THEORY:

BCH Code:

The BCH code are the most powerful and widely used random error correcting cyclic codes. These code were discovered by Hocquenghem in 1959 and independently by Bose and Choudhary in 1960.

An (n,k) binary BCh code is specified as,

Block length $n= 2^m – 1$

Parity check bits n-k = mtc

Minimum distance dmin ≥ 2 tc +1

Where m ≥ 3 is any integer and tc is no. of error the code is capable of correcting.

Procedure :

1. Given code length n and error correcting capability tc
2. Find m = log2 (n + 1) where $n= 2^m – 1$
3. Find $k = 2^m – 1 – mtc = n - mtc$
4. Take a primitive polynomial of degree m
5. Construct the finite field $GF(2^m)$ using the primitive polynomial p(x)
6. Find the minimal polynomial of each element in $GF(2^m)$

7. Find g(x) = LCM [m1(x).m3(x). m5(x)…….m2tc-1(x)]
8. Find the code word using c(x) = d(x) . g(x)

Conclusion:

Program:

```
% Implementation of algorithms for decoding of BCH algorithm.
%BCH Encoding
clear all;
clc;
m =input ('m=') ;
fprintf('Codeword length ')
n = 2^m-1 % Codeword length
k = input ('enter message length k='); % Message length
m=input('enter the message of the length m=')
msg = gf(m)
% Find t, the error-correction capability.
[genpoly,t] = bchgenpoly(n,k);
disp('Error correcting capability :');
disp(t);
%t2=input('Enter no. of errors to be added to produce noisy code :');
% Encode the message.
code = bchenc(msg,n,k)
% Corrupt up to t2 bits in each codeword.
tn=1*6;
noisycode = code + randerr(1,n,1:tn)
% Decode the noisy code.
[newmsg,err,ccode] = bchdec(noisycode,n,k);
fprintf('new message=')
disp(newmsg)
```

```
if msg==newmsg
disp('The message was recovered perfectly.')
else
disp('Error in recovery of message.')
end;

% Implementation of algorithms for RS Coding & Decoding
%RS CODING
clear all;
clc;
n=input('accept n=');
k=input('accept k=');
m=input('accept message=');
msg=gf([m],k);
c = rsenc(msg,n,k); % Code will be a Galois array.


%RS DECODING
clear all; clc;
n=input('accept n=');
k=input('accept k=');
m=input('accept message=');
msg=gf([m],k);
c = rsenc(msg,n,k) % Code will be a Galois array.
r=c;
r(1)=2;
r(3)=1;
r(10)=29;
r(11)=12;
r(12)=18;
[d,e]=rsdec(r,n,k)
```