Information Theory and Codes

Homeworks

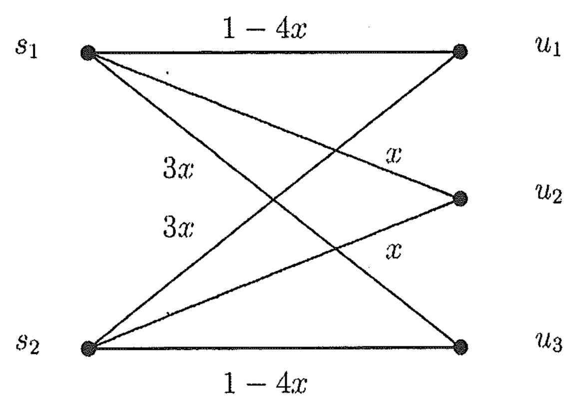
Ferro Demetrio	207872
Minetto Alex	211419

Problem 1

Compute the maximum capacity (with respect to x) of a symmetric channel whose stochastic matrix depends on the parameter x

$$\begin{bmatrix} 1-4x & x & 3x \\ 3x & x & 1-4x \end{bmatrix}$$

By observing the matrix, we can see that the channel has two input and three possible output, so it can be modeled as a BEC channel whose Source set of symbols is $S = \{s_i, i = 1, 2\}$ and User's set of symbols is $U = \{u_j, j = 1, 2, 3\}$.



In order to evaluate the capacity, we start considering the probability mass function of the input symbols.

Let us suppose to transmit the first symbol with probability $P(s = s_1) = p$, and the second one with probability $P(s = s_2) = (1 - p)$, where $p \in [0, 1]$.

The channel stochastic characterization can be easily represented by the following:

$$P(s) = \begin{bmatrix} p \\ 1-p \end{bmatrix}, \qquad P(u|s) = \begin{bmatrix} 1-4x & x & 3x \\ 3x & x & 1-4x \end{bmatrix}$$

By the definition of conditional probability,

$$P(u|s) = \frac{P(s,u)}{P(s)} \qquad P(s,u) = P(u|s)P(s)$$

We obtain the joint distribution:

$$P(s,u) = \{P(s = s_i, u = u_j), i = 1, 2, j = 1, 2, 3\}$$

$$P(s,u) = \begin{bmatrix} (1-4x)p & xp & 3xp \\ 3x(1-p) & x(1-p) & (1-4x)(1-p) \end{bmatrix}$$

We may get to $P(u)$ by summing over the columns.

$$P(u) = \sum_{s_i \in S} P(s = s_i, u) \qquad P(u) = \begin{bmatrix} (1-4x)p + 3x(1-p) \\ xp + x(1-p) \\ 3xp + (1-4x)(1-p) \end{bmatrix}$$

That's what we get by multiplying the two matrices $P(u|s)$ and $P(s)$.

Given that we have a free parameter x, we have to evaluate the range in which we want to let such variable vary.

In order to get probabilities, the following has to be verified:

1) $0 \leq P(u|s) \leq 1$, $\forall s \in S, u \in U$.

$$\begin{array}{l} 0 \leq 1 - 4x \leq 1 \\ 0 \leq x \leq 1 \\ 0 \leq 3x \leq 1 \end{array}$$

which gives us $x \in [0, \frac{1}{4}] \subset \mathbb{R}$.

2) $\sum_{s_i \in S} \sum_{u_j \in U} P(s = s_i, u = u_j) = \sum_{u_j \in U} P(u = u_j) = 1$
that's always true for every $x \in [0, \frac{1}{4}] \subset \mathbb{R}$ and for every $p \in [0, 1] \subset \mathbb{R}$.

Then we compute the mutual information between the source and the user.

$C(x) = \max_{P(s)} I(u, s)$

$$I(u, s) = \sum_{s_i \in S} \sum_{u_j \in U} P(s, u) \log_2 \left(\frac{P(s, u)}{P(s)P(u)} \right) = \sum_{s_i \in S} \sum_{u_j \in U} P(s, u) \log_2 \left(\frac{P(u|s)}{P(u)} \right)$$

Since we had $P(s, u)$ since before, we computed:

$$\frac{P(u|s)}{P(u)} = \begin{bmatrix} \frac{1-4x}{p-7xp+3x} & 1 & \frac{3x}{7xp-4x-p+1} \\ \frac{3x}{p-7xp+3x} & 1 & \frac{1-4x}{7xp-4x-p+1} \end{bmatrix}$$

Then we discussed the singularities of the matrix:

$$p - 7xp + 3x \neq 0 \qquad x \neq \frac{p}{(7p-3)} \qquad \begin{array}{l} p = 0 \rightarrow x \neq 0 \\ p = 1 \rightarrow x \neq \frac{1}{4} \end{array}$$

$$7xp - 4x - p + 1 \neq 0 \qquad x \neq \frac{p-1}{7p-4} \qquad \begin{array}{l} p = 0 \rightarrow x \neq \frac{1}{4} \\ p = 1 \rightarrow x \neq 0 \end{array}$$

Yields to singularities for the mutual information in ($x = 0; x = \frac{1}{4}$) so we will consider $x \in]0, \frac{1}{4}[\subset \mathbb{R}$.

In order to compute the capacity of the channel, in function of x and of the parameter p, we wrote a Matlab code:

```
clc
clear all
close all

x=sym('x','real');
p=sym('p','real');
Pxy=sym('Pxy','real');
Py_x_y=sym('Py_x_y','real');

Py_x=[ 1-4*x x 3*x; 3*x x 1-4*x];

Px=[p; 1-p];

Py=Py_x'*Px;

for i=1:size(Py_x,1)
    for j=1:size(Py_x,2)
        Pxy(i,j)=Py_x(i,j)*Px(i);
        Py_x_y(i,j)=Py_x(i,j)/Py(j);
    end
end

Py_s=[p-7*p*x+3*x; x; 7*p*x-p-4*x+1];

Hx=sum(Px.*log2(1./Px));
Hx_ev=[];

Hy=sum(Py_s.*log2(1./Py_s));
Hy_ev=[];

Ixy=sum(sum(Pxy.*log2(Py_x_y)));
Ixy_ev=[];


x=0:.01:.25;
px=0:.1:1;

for i=1:11
if(px(i)==0)
    px(i)=1e-10;
end
if(px(i)==1)
    px(i)=1-1e-10;
end

if(x(1)==0)
    x(1)=1e-10;
end
if(x(end)==0.25)
    x(end)=0.25-1e-10;
end

p=px(i);
Px_ev=eval(Px);
Hx_ev=[Hx_ev; eval(Hx)];

subplot(2,11,i)
stem([0,1],Px_ev,'fill'), hold on
axis([-0.5 1.5 0 1])
```

```
xlabel(sprintf('p_0=%2.2f\np_1=%2.2f',p,(1-p)));

Py_s_ev=eval(Py_s);
Hy_ev=[Hy_ev; eval(Hy)];

subplot(2,11,i+11)
stem([0,0.5,1],Py_s_ev,'fill'), hold on
axis([-0.5 1.5 0 1])

Ixy_ev=[Ixy_ev; eval(Ixy)];
end

[Itrmax, Xindexmax]=max(Ixy_ev');
[C, Pxindexmax]=max(Ixy_ev);
Cmax=Ixy_ev(Pxindexmax(1),Xindexmax(1))

legendCell = cellstr(num2str(x', 'x=%2.2f'));
h=legend(legendCell);
set(h,'FontSize',8,'Location','EastOutside');

f2=figure(2);
set(f2,'Name','Entropy of the source and of the user')
subplot(2,1,1)
plot(px,Hx_ev,'r'), hold on
title('Entropy of the source');
ylabel('H(s)');
xlabel('p = [0, 1]');
subplot(2,1,2)
plot(px,Hy_ev')
legendCell1 = cellstr(num2str(x', 'x=%2.2f'));
h1=legend(legendCell1);
set(h1,'FontSize',8,'Location','EastOutside');
title('Entropy of the user');
ylabel('H(u)');
xlabel('p = [0,1]');

f3=figure(3);
set(f3,'Name','Mutual information between Source and User')
plot(x,Ixy_ev)
hold on
plot(x(Xindexmax(1)),Cmax(1),'r.')
hold on
legendCell2 = cellstr(num2str(px', 'p=%2.2f'));
h2=legend(legendCell2);
set(h2,'FontSize',8,'Location','EastOutside');
title('Mutual information between Source and User');
text(x(Xindexmax(1))+0.003,Cmax(1)-0.03, sprintf('Cmax=%2.2f',Cmax(1)), 'VerticalAlignment','
    bottom', ...
                                'HorizontalAlignment','left')
ylabel('Ixy');
xlabel('x = [0, 1/4]');

f4=figure(4);
set(f4,'Name','Mutual information behaviour in fuction of p and x')
surf(x,px,Ixy_ev)
hold on
title('Mutual information behaviour in fuction of p and x');
xlabel('x = [0, 1/4]');
ylabel('p = [0, 1]');
zlabel('I(u,s)');
```

Exercise 1 implementation in Matlab.

Whose output is the following:

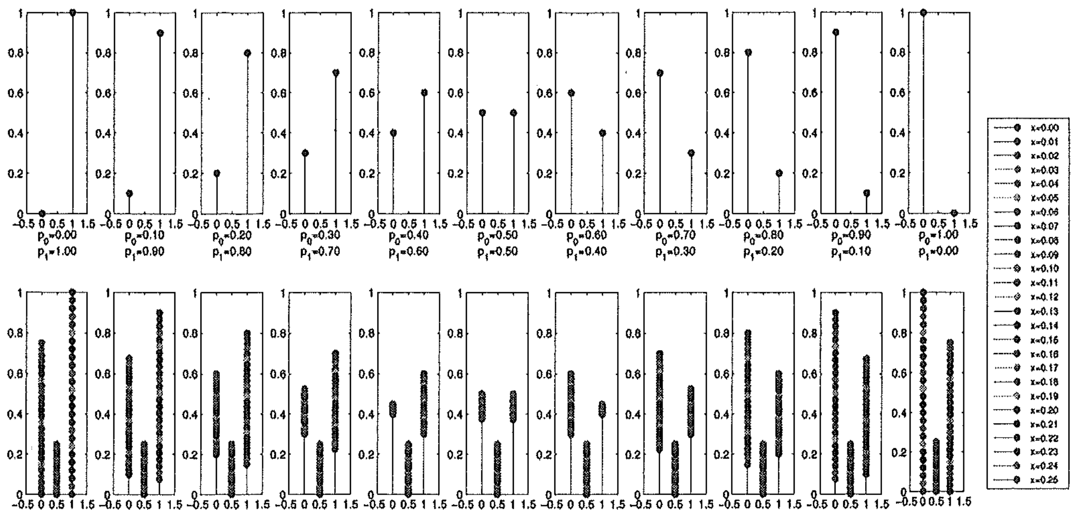


Figure 1: Probability mass functions of the Source and of the User in function of $p \in [0,1]$ and $x \in]0, \frac{1}{4}[$.

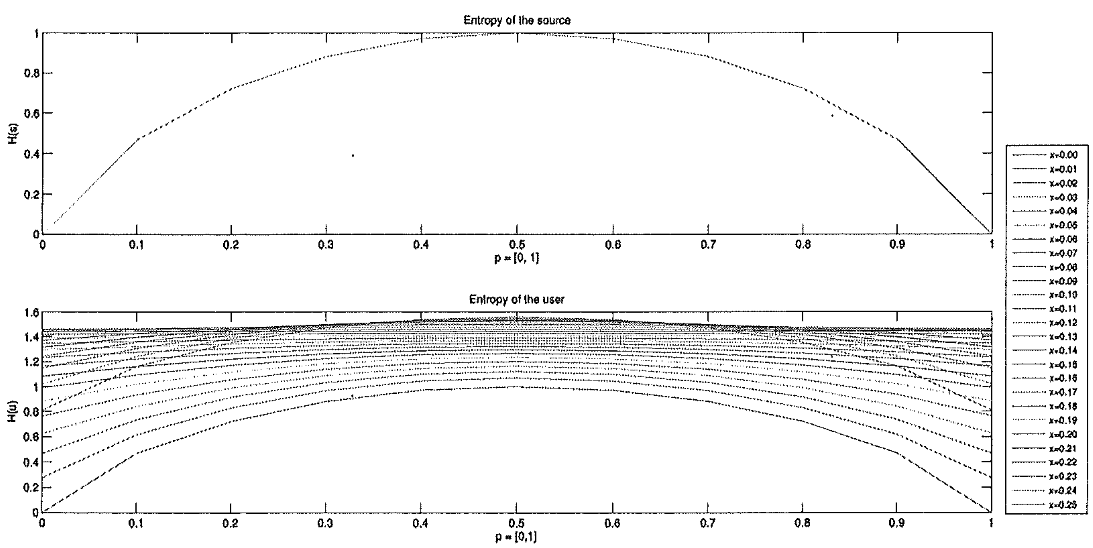


Figure 2: Entropy of the Source and of the User in function of $p \in [0,1]$ and $x \in]0, \frac{1}{4}[$.

At first we noticed that the Channel Capacity: $C(x) = \max_{P(s)} I(s,u)$ is achieved, for every x, in $P(s) = [\frac{1}{2}, \frac{1}{2}]$, so when $p = 1 - p = \frac{1}{2}$.

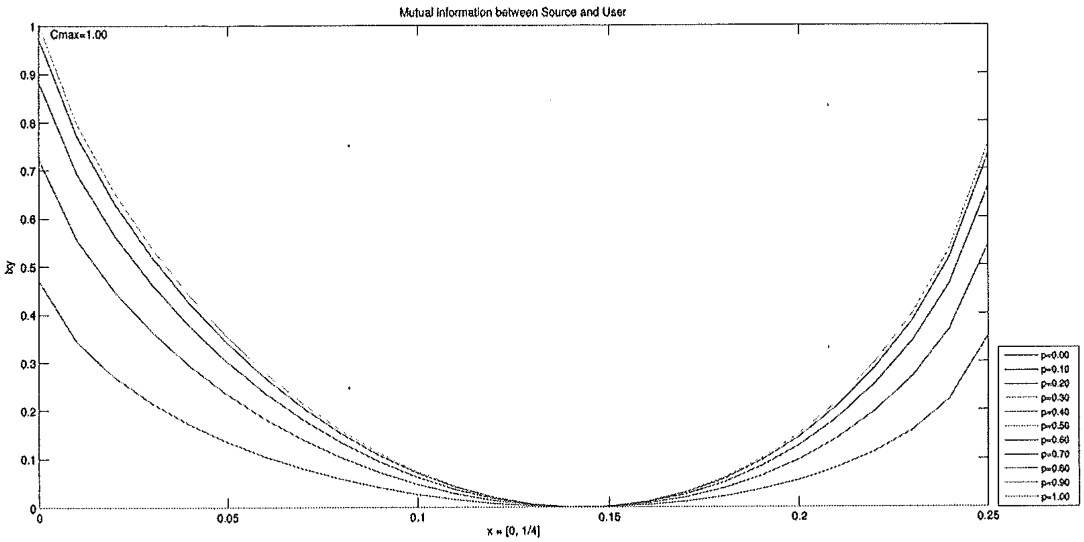


Figure 3: Mutual Information between the Source and the User in function of $p \in [0,1]$ and $x \in]0, \frac{1}{4}[$.

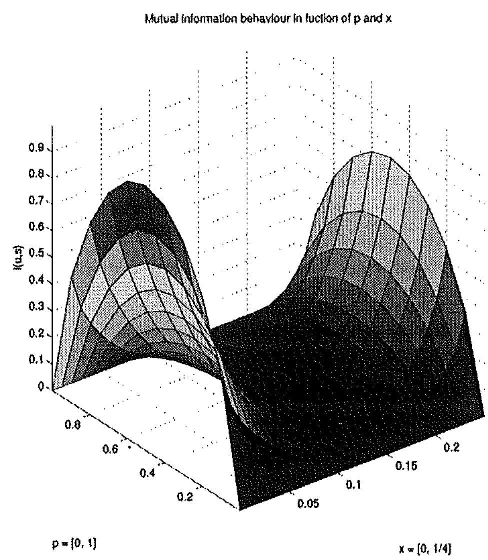


Figure 4: Mutual Information between the Source and the User in function of $p \in [0,1]$ and $x \in]0, \frac{1}{4}[$.

The maximum Capacity in function of x is: $C_{\max} = \max_{x \in]0,\frac{1}{4}[} C(x)$ and it's maximum when $x \to 0$ from the right. We may consider, then: $\lim_{x \to 0^+} C_{\max}(x) = 1$.
because, in our channel, for values very close to zero, the channel has in ideal behaviour, its error probability is close to zero.

It can be seen easily by considering $x = 0$:

$$P(u|s)\Big|_{x=0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad\qquad P(e) = \sum_{s_i \in S} P(u = s_i | s \neq s_i) = 0$$

Yields to the maximum capacity, since we are transmitting as much amount of information as possible, without any error.

Afterwards, we noticed that in $x = \frac{1}{7}$ the channel has zero capacity.

This may be easily seen by considering the ratio between the conditional probability and the User's symbol probability.

$$\frac{P(u|s)}{P(u)}\Big|_{x=\frac{1}{7}} = \begin{bmatrix} \frac{1-\frac{4}{7}}{p-p+\frac{3}{7}} & 1 & \frac{\frac{3}{7}}{p-\frac{4}{7}-p+1} \\ \frac{\frac{3}{7}}{p-p+\frac{3}{7}} & 1 & \frac{1-4x}{7xp-4x-p+1} \end{bmatrix}_{x=\frac{1}{7}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In order to get to a deeper view, let us think to what the ratio stands for:

$$\frac{P(u|s)}{P(u)} = \frac{P(u|s)P(s)}{P(u)P(s)} = \frac{P(s,u)}{P(u)P(s)} = 1, \;\; \forall s_i \in S, u_j \in U \quad \rightarrow \quad P(s,u) = P(u)P(s)$$

The product of the probability mass function of source symbols and the probability mass function of the user's symbols is equal to the joint probability function!

In other words, $P(s,u) = P(u)P(s)$ means that $P(u)$ does not depend on $P(s)$, then what you get at the end of the channel does not depend on what you transmit anymore.

Moreover, $\quad I(u,s)\Big|_{x=\frac{1}{7}} = \sum_{s_i \in S} \sum_{u_j \in U} P(s,u)\Big|_{x=\frac{1}{7}} \log_2(1) = 0, \qquad \forall s_i \in S, \forall u_j \in U.$

Problem 2

An alphabet consists of five digits $\{1, 2, 3, 4, 5\}$ which are used with the same probability. Find the average length of the code words of an optimal Huffman binary code considering words of two digits. Compute the efficiency of the code (i.e. the ratio between the enetropy of the primary alphabet and the average length of the secondary words per symbol), and compare this efficiency with that of a binary block code that encodes the words of two digits in blocks of 6 bits.

In order to find the average length of the Huffman code for an alphabet of N digits mapped in words of k digits, we have to compute all the possible combinations, consisting in N^k words.

Since the digits of the primary alphabet are equiprobable with probability $\frac{1}{N}$, we will get equiprobabile words equiprobable with probability $\left(\frac{1}{N}\right)^k$.

In our alphabet we have 5 digits $X = x_i, i = 1, ..., 5$, equally distributed with probability $P(X = x_i) = \frac{1}{5}, \quad \forall i = 1, ..., 5$.

The alphabet built up considering words of two digits will have 25 words $Y = y_j, j = 1, ..., 25$, equally distributed with probability $P(Y = y_j) = \frac{1}{25}, \quad \forall j = 1, ..., 25$.

Huffman coding requires to build up a binary tree.

In order to do that, we wrote a Matlab code:

```
clear all
close all
clc

Nsymbols=5;
Ndigits=2;

M=Nsymbols^Ndigits;

if  mod(log2(M),1)==0
    Depth=floor(log2(M));
    Ndots=2^(Depth+1)-1;
else
    Depth=floor(log2(M))+1;
    Ndots=2*((2^(Depth-1)-(2^Depth-M)).*2+2^Depth-M)-1;
end

Stepx=2^Depth;

TotDotsR=zeros(1,Depth+1);
TotDotsL=zeros(1,Depth+1);

rx=zeros(1,Ndots);
ry=1;
qy=0;
k=1;
kr=1;
kl=1;
ryprev=0;
ddots=0;


figure(1)
```

```
set(gcf,'name',sprintf('Huffman Tree: %d symbols',M),'numbertitle','off');

for i=1:Ndots
    h=waitbar(i/Ndots);
    ry=floor(log2(i));

    if(qy~=ry)
      Stepx=Stepx/2;
      k=k+1;
    end

    if i==1
      rx(i)=0;
    elseif i==Ndots && mod(TotDotsL(k)+1,2)~=0
      rx(i)=Stepx*(i+2);
      TotDotsR(k)=TotDotsR(k)+1;
    elseif(mod(i,2)==0)
      rx(i)=Stepx*(i+1);
      TotDotsR(k)=TotDotsR(k)+1;
    else
      rx(i)=-Stepx*i;
      TotDotsL(k)=TotDotsL(k)+1;
    end
    qy=ry;
    plot(rx(i),ry,'k.','markersize',12)
    % axis off
    if(i>=M)
    text(rx(i)-0.1,ry+0.2,strcat(strcat('X_{',num2str(i-M+1),'}')));
    end
    title(sprintf('Huffman Tree: %d symbols',M));
    hold on
    if(ry>1 &&  i<Ndots)
        ddots=mod(i,4);
        if ddots < 2
            plot([rx(i) rx(round(i/2))],[ry ry-1],'k')
            hold on
        else
            plot([rx(i) rx(round(i/2)-1)],[ry ry-1],'k')
            hold on
        end
    elseif (i>1 && i<4) || (i==Ndots)
        plot([rx(i) rx(floor(i/2))],[ry ry-1],'k')
        hold on
    end
end
close(h);

TotDots=TotDotsL+TotDotsR;

if mod(TotDots(end),2)==0
    TDots=0.5*TotDots(end)+TotDots(end-1);
else
    TDots=floor(0.5*TotDots(end))+TotDots(end-1);
end

Lavg=(TotDots(end)*Depth+(2^Depth-M)*(Depth-1))/M;

Lavg1=Lavg/Ndigits;

Entropy_Ndigits=1/Ndigits*log2(M);
```

Exercise 2 implementation in Matlab.

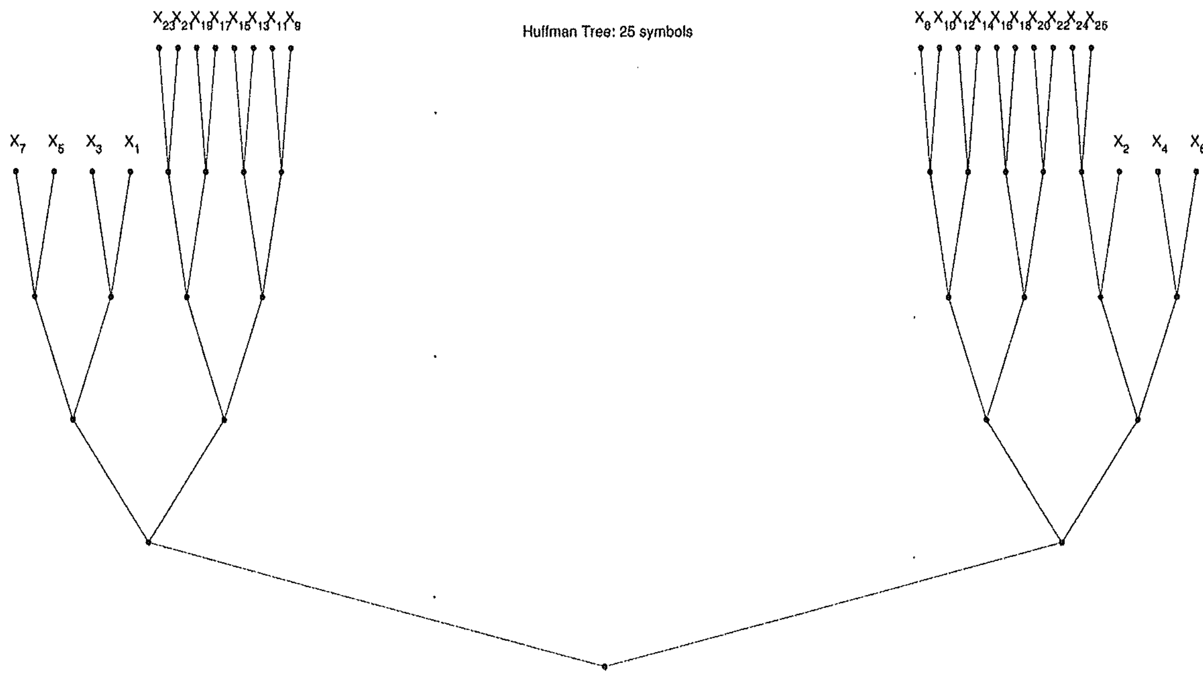


Figure 5: Huffman Tree built on the set of words of two symbols belonging to a 5 digits alphabet.

By this way, we may compute the average length of the code words:

$$\bar{\ell} = \frac{7 \cdot 4 + 18 \cdot 5}{25} = 4.72 \text{ bits}$$

In order to evaluate the efficiency, we may compute the Entropy of the secondary alphabet:

$$H(Y) = \sum_{y_j \in Y} P(y) \log_2 \left(\frac{1}{P(y)} \right)$$

$$H(Y) = \sum_{i=1}^{25} \frac{1}{25} \log_2(25) = 2 \log_2(5) = 4.64 \text{ bits/use.}$$

Then we compute the efficiency of the code:

$$\eta_1 = \frac{H(Y)}{\bar{\ell}} = \frac{2 \log_2 5}{4.72} = 0.98;$$

Then we proceed encoding the words of two digits in blocks of 6 digits.

Let us consider first the fixed block coding of the primary symbols:

1	000
2	001
3	010
4	011
5	100

Words of two digits will be encoded considering the combination of couples of the previous symbols:

1 1	000 000
1 2	000 001
1 3	000 010
1 4	000 011
1 5	000 100
2 1	001 000
2 2	001 001
2 3	001 010
2 4	001 011
2 5	001 100
3 1	010 000
3 2	010 001
3 3	010 010
3 4	010 011
3 5	010 100
4 1	011 000
4 2	011 001
4 3	011 010
4 4	011 011
4 5	011 100
5 1	100 000
5 2	100 001
5 3	100 010
5 4	100 011
5 5	100 100

In this case the length of the words is constant and equal to 6.

$\overline{\ell} = 6$ bits

We may evaluate the efficiency of the coding by considering again:

$$\eta_2 = \frac{H(Y)}{\overline{\ell}} = \frac{2\log_2(5)}{6} = 0.77;$$

In conclusion, we have that extended Huffman coding with words of two digits is more efficient than binary block coding that encodes words of two digits, since $\eta_1 > \eta_2$.

Problem #3

A message has been represented as a binary stream, with letters (blank and punctuation signs counting for letters) represented by blocks of 5 bits according to the table loaded in the Web site, folder Materiale.
The message has been encoded with a non-systematic cyclic code (15,5,7) and sent on a noisy channel. The code generator polynomial is:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

Decode the following received message:

1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1,
1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1; 0, 1, 0, 0, 1, 1, 1, 0,
1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1; 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0

The whole encoded message can be found in the Web site, folder Materiale, file **EncodedMessageHom2**

The algebraic decoding may be performed working in the finite field $\mathbb{F}(2^4)$ with generator polynomial:

$$m(x) = x^4 + x + 1.$$

In order to decode the message, we considered the roots of the generator polynomial $m(x)$ in the extended field $\mathbb{F}_{2^4}$.

$$\begin{array}{ll||ll||ll}
\alpha^0 & 1 & \alpha^5 & \alpha+\alpha^2 & \alpha^{10} & 1+\alpha+\alpha^2 \\
\alpha^1 & \alpha & \alpha^6 & \alpha^2+\alpha^3 & \alpha^{11} & \alpha+\alpha^2+\alpha^3 \\
\alpha^2 & \alpha^2 & \alpha^7 & 1+\alpha+\alpha^3 & \alpha^{12} & 1+\alpha+\alpha^2+\alpha^3 \\
\alpha^3 & \alpha^3 & \alpha^8 & 1+\alpha^2 & \alpha^{13} & 1+\alpha^2+\alpha^3 \\
\alpha^4 & 1+\alpha & \alpha^9 & \alpha^2+\alpha^3 & \alpha^{14} & 1+\alpha^3
\end{array}$$

For every sequence of 15 bits received: $[r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14},]$, we evaluate the corresponding polynomial:

$$\begin{aligned} r(x) = r_0 + r_1x + r_2x^2 + r_3x^3 + r_4x^4 + r_5x^5 + r_6x^6 + r_7x^7 + \\ + r_8x^8 + r_9x^9 + r_{10}x^{10} + r_{11}x^{11} + r_{12}x^{12} + r_{13}x^{13} + r_{40}x^{14}. \end{aligned}$$

Since the polynomial obtained corresponds to the received word, transmitted on a channel affected by noise, $r(x) = c(x) + e(x)$, where $c(x)$ and $e(x)$ are respectively the code word and the error polynomials.

The encoding rule allows to recover up to 3 errors, given that $d = 7$, $d = 2t + 1$, where $t = 3$ is the maximum number of correctable errors.

By considering $r(x) = c(x) + e(x)$, where $c(x) = g(x) \cdot I(x)$ is the product of the generator polynomial $g(x)$ and the information polynomial $I(x)$ and $e(x) = E_1x^{j_1} + E_2x^{j_2} + E_3x^{j_3}$.

But, being in a $\mathbb{F}_2$ field, the magnitude of the error $E_i \in \{0, 1\}, \forall i = 1, 2, 3$.

The decoding procedure was achieved by using the Peterson-Gorenstein-Zierler algorithm (PGZ). We computed $2t = 6$ syndromes: $S_1, S_2, S_3, S_4, S_5, S_6$ by evaluating the received polynomial in the roots of the generator polynomial: $S_k = r(\alpha^k), \forall k = 1, \cdots, 2t$.

$$\left\{\begin{array}{l} S_1 = c(\alpha) + e(\alpha) \\ S_2 = c(\alpha^2) + e(\alpha^2) \\ S_3 = c(\alpha^3) + e(\alpha^3) \\ S_2 = c(\alpha^4) + e(\alpha^4) \\ S_5 = c(\alpha^5) + e(\alpha^5) \\ S_6 = c(\alpha^6) + e(\alpha^6) \end{array}\right. = \left\{\begin{array}{l} S_1 = \alpha^{j_1} + \alpha^{j_2} + \alpha^{j_3} \\ S_2 = \alpha^{2j_1} + \alpha^{2j_2} + \alpha^{2j_3} \\ S_3 = \alpha^{3j_1} + \alpha^{3j_2} + \alpha^{3j_3} \\ S_2 = \alpha^{4j_1} + \alpha^{4j_2} + \alpha^{4j_3} \\ S_5 = \alpha^{5j_1} + \alpha^{5j_2} + \alpha^{5j_3} \\ S_6 = \alpha^{6j_1} + \alpha^{6j_2} + \alpha^{6j_3} \end{array}\right.$$

By considering $r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i) = 0\ \forall i = 1, ..., 6$, since $c(\alpha_i) = 0 \forall i = 1, ..., 6$.

The next step was the implementation of an Error Locator Polynomial.
First of all, we noticed that there could be a different number of errors, running from 1 to 3.

In order to find the number of errors present in the received message, let us consider that ν is the actual number of errors, where $\nu \in \{1, 2, 3\}$.
The error locator polynomial will be different depending on the number of errors ν:

$$\sigma^{(\nu)}(z) = z^\nu + \sigma_1 z^{\nu-1} + ... + \sigma_\nu$$

In order to find the solutions of the error locator polynomial, we may rewrite it differently:

$$\sigma^{(\nu)}(z) = \prod_{i=1}^{\nu}(z + z_i)$$

Now, we want $\{z_j\}_{j=1}^{\nu}$ to be the roots of the polynomial.

$$z_j^{\nu} + \sigma_i z_j^{\nu-1} + ... + \sigma_{\nu} = 0$$

Now, for $t = 3$ we will have:

$$\begin{cases} z_1^3 + \sigma_1 z_1^2 + \sigma_2 z_1 + \sigma_3 = 0 \\ z_2^3 + \sigma_1 z_2^2 + \sigma_2 z_2 + \sigma_3 = 0 \\ z_3^3 + \sigma_1 z_3^2 + \sigma_2 z_3 + \sigma_3 = 0 \end{cases} = \begin{cases} \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 = -S_4 \\ \sigma_1 S_4 + \sigma_2 S_3 + \sigma_{\nu} S_2 = -S_5 \\ \sigma_1 S_5 + \sigma_2 S_4 + \sigma_3 S_3 = -S_6 \end{cases}$$

which can be written easier this way:

$$\begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} \cdot \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_4 \\ -S_5 \\ -S_6 \end{bmatrix}$$

In general, we may say that there will always be:

$$\begin{bmatrix} S_1 & \cdots & S_{\mu} \\ \vdots & \ddots & \vdots \\ S_{\mu} & \cdots & S_{2\mu-1} \end{bmatrix} \cdot \begin{bmatrix} \sigma_{\mu} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{\mu+1} \\ -S_{\mu+2} \\ -S_{2\mu} \end{bmatrix}$$

Call $\mathbf{M}_{\mu}$ the first matrix containing all the $(2\mu - 1)$ syndromes.

It can be shown that, if there are $\nu \leq 3$ errors, then the matrix $\mathbf{M}_{\mu}$ is nonsingular, but $\mathbf{M}_{\mu}$ is singular if there are $\mu > \nu$ errors.

Finding the largest $\nu \leq t$ such that $\det(\mathbf{M}_{\nu}) \neq 0$, we get the number of errors.

So we may compute the $\{\sigma_j\}_{j=1}^{\nu}$ coefficients of the error locator polynomial.

Now, in order to find the position of the error, we have to evaluate the polynomial in all the roots of the extended field.

$$\sigma^{(\nu)}(\alpha^i) \begin{cases} = 0 & \text{if there is an error in position } i; \\ \neq 0 & \text{if there isn't any error in position } i; \end{cases} \quad \forall i = 0, ..., 14$$

Now that we know how many errors are present (ν), and which are their positions $\{j_i\}_{i=1}^{\nu}$, we may estimate the error polynomial:

$$\hat{e}(x) = \sum_{i=1}^{\nu} x^{j_i}$$

And the correct code word that we should receive:

$$\hat{c}(x) = r(x) + \hat{e}(x).$$

Now, by dividing it by $g(x)$, we get to the actual information message $I(x)$.

$$I(x) = \frac{\hat{c}(x)}{g(x)}$$

which will be mapped again in 5 bits, which will correspond to letters with a suitable function that we developed, which will be shown further.

The correct message was the following:
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
0, 1, 1, 1, 1, 1, 0, 0,

which corresponds to the string:

"if we will be quiet and ready enough,we shall find compensation in every disappointment."

That's the coded one.

Here is reported the code developed to solve the assigned problem:

```
clear all
close all
clc

fdrxmsg =fopen('receivedmessage.txt','r');

RxMsg=textscan(fdrxmsg,'%d','delimiter', ',');

RxMsg=RxMsg{1,1};

syms x g_x m_x I_x_bin_tot alpha roots roots_ev

I_x_bin_tot=[];
n_err_tot=[];

m_x=x^4+x+1;

roots=[alpha^12 alpha^11 alpha^13 alpha^10 alpha^7 ...
       alpha^14 alpha^9 alpha^8 alpha^6 alpha^5 alpha^4];

roots_ev=[1+alpha+alpha^2+alpha^3 ...
          alpha+alpha^2+alpha^3 ...
          1+alpha^2+alpha^3 ...
          1+alpha+alpha^2 ...
          1+alpha+alpha^3 ...
          1+alpha^3 ...
          alpha+alpha^3 ...
          1+alpha^2 ...
          alpha^2+alpha^3 ...
          alpha+alpha^2 ...
          1+alpha];

g_x=x^10+x^8+x^5+x^4+x^2+x+1;

wordIndex=0;

fprintf('Starting to decode ...\n');

for wordIndex=1:15:length(RxMsg)

syms r_x c_x e_x I_x I_x_bin  ...
     S1 S2 S3 S4 S5 S6 z sigma_z sigma1 sigma2 sigma3

RxWord=RxMsg(wordIndex:wordIndex+14);

% COMPUTE THE SYNDROMES
r_x=poly2sym(RxWord(end:-1:1));

S1=subs(r_x,x,alpha);
S2=subs(r_x,x,alpha^2);
S3=subs(r_x,x,alpha^3);
S4=subs(r_x,x,alpha^4);
S5=subs(r_x,x,alpha^5);
S6=subs(r_x,x,alpha^6);

[q1,S1]=quorem(S1,subs(m_x,x,alpha));
[q2,S2]=quorem(S2,subs(m_x,x,alpha));
[q3,S3]=quorem(S3,subs(m_x,x,alpha));
[q4,S4]=quorem(S4,subs(m_x,x,alpha));
```

```
[q5,S5]=quorem(S5,subs(m_x,x,alpha));
[q6,S6]=quorem(S6,subs(m_x,x,alpha));

S1=mod(S1,2);
S2=mod(S2,2);
S3=mod(S3,2);
S4=mod(S4,2);
S5=mod(S5,2);
S6=mod(S6,2);

if mod(subs(mod(det([S1, S2, S3; S2, S3, S4; S3, S4, S5]),2),roots,roots_ev),2)==0
    if mod(subs(mod(det([S3, S4; S4, S5]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S2, S4; S3, S5]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S2, S3; S3, S4]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S2, S3; S4, S5]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S1, S3; S3, S5]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S1, S2; S3, S4]),2),roots,roots_ev),2)==0 ...
    || mod(subs(mod(det([S3, S4; S4, S5]),2),roots,roots_ev),2)==0
       n_err=1;
    else
       n_err=2;
    end
else
       n_err=3;
end
n_err_tot=[n_err_tot, n_err];
switch(n_err)
case 1
 sigma_z=z+S1;

case 2
 Eq1=S3+sigma1*S2+sigma2*S1;
 Eq2=S4+sigma1*S3+sigma2*S2;

 [sigma1, sigma2]=solve(Eq1, Eq2);

 sigma1=subs(sigma1, roots, roots_ev);
 [sigma1num, sigma1den]=numden(sigma1);
 sigma1num=mod(sigma1num,2);
 sigma1den=mod(sigma1den,2);
 sigma1=subs(simplify(sigma1num/sigma1den),roots_ev,roots);

 sigma2=subs(sigma2, roots, roots_ev);
 [sigma2num, sigma2den]=numden(sigma2);
 sigma2num=mod(sigma2num,2);
 sigma2den=mod(sigma2den,2);
 sigma2=subs(simplify(sigma2num/sigma2den),roots_ev,roots);

 sigma_z=z^2+sigma1^2*z+sigma2;

case 3
 Eq1=-S4+sigma1*S3+sigma2*S2+sigma3*S1;
 Eq2=-S5+sigma1*S4+sigma2*S3+sigma3*S2;
 Eq3=-S6+sigma1*S5+sigma2*S4+sigma3*S3;

 [sigma1, sigma2, sigma3]=solve(Eq1, Eq2, Eq3);

 sigma1=subs(sigma1, roots, roots_ev);
 [sigma1num, sigma1den]=numden(sigma1);
 sigma1num=mod(sigma1num,2);
 sigma1den=mod(sigma1den,2);
 sigma1=subs(simplify(sigma1num/sigma1den),roots_ev,roots);
```

```
  sigma2=subs(sigma2, roots, roots_ev);
  [sigma2num, sigma2den]=numden(sigma2);
  sigma2num=mod(sigma2num,2);
  sigma2den=mod(sigma2den,2);
  sigma2=subs(simplify(sigma2num/sigma2den),roots_ev,roots);

  sigma3=subs(sigma3, roots, roots_ev);
  [sigma3num, sigma3den]=numden(sigma3);
  sigma3num=mod(sigma3num,2);
  sigma3den=mod(sigma3den,2);
  sigma3=subs(simplify(sigma3num/sigma3den),roots_ev,roots);

  sigma_z=z^3+sigma1*z^2+sigma2*z+sigma3;
end

e_x=0;

for j=0:14
    sigma_z_ev=subs(sigma_z,z,alpha^j);
    [sigmanum, sigmaden]=numden(sigma_z_ev);
    modsigmanum=mod(sigmanum,2);
    [sigmaquot, sigmarem]=quorem(modsigmanum,subs(m_x,x,alpha));
    modsigma=mod(expand(sigmarem),2);
    if(modsigma==0)
        e_x=e_x+x^j;
    end
end
c_x=mod(e_x+r_x,2);
[I_x, I_x_rem]=quorem(c_x,g_x);
I_x=mod(I_x,2);
I_x_bin=sym2poly(I_x);
tofit=5-length(I_x_bin);
for i=1:tofit
    I_x_bin=[0,I_x_bin];
end
I_x_bin_tot=[I_x_bin,I_x_bin_tot];
clc;
fprintf('Decoding in progress: %2.0f%c\n',(wordIndex+15)/length(RxMsg)*100,'%')
end

clc
fprintf('Decoding complete:\n');

fdcorrectmsg =fopen('correctmessage.txt','w');

fprintf(fdcorrectmsg,'%d, ',I_x_bin_tot(end:-1:1));

[IndexRes, StrRes] = MappingToString('correctmessage.txt', 5,'alphabetmapping.txt');
disp(StrRes);

fclose('all');
```

Problem solution implementation in Matlab.

which does exactly what was theoretically formulated in the previous pages.

The following is a function which maps the coded 5 bits in letters:

```
function [IndexRes, StrRes] = MappingToString(RxMessage, Step, AlphabetMapping)

fdrxmsg =fopen(RxMessage,'r');
fdabmap =fopen(AlphabetMapping,'r');

RxMsg=textscan(fdrxmsg,'%d','delimiter', ',');

AlphaBetMap=textscan(fdabmap,'%c %s','delimiter', '_');

RxMsg=RxMsg{1,1};

AlphaBetMap=AlphaBetMap{1,1};
StrRes='';

IndexRes=[];


for i=1:Step:length(RxMsg)
    Index=0;

    for j=0:Step-1
    Index=Index+2^(Step-1-j)*RxMsg(j+i);
    end

    if Index==26
        StrRes=strcat(StrRes,{' '});
    else
        StrRes=strcat(StrRes,AlphaBetMap(Index+1));
    end

    IndexRes=[IndexRes, Index];
end

fclose('all');

end
```

Mapping to string implementation in Matlab.