Ψηφιακές Τηλεπικοινωνίες

Εργαστηριακή Άσκηση Σετ 1

Ανδρονίκη Πορτοκάλογλου
1067539
4ο Έτος

Περιεχόμενα

1.Ερώτημα Πρώτο-Κωδικοποίηση Huffman

1.1

1.2

1.3

1.4

1.5

2.Ερώτημα Δεύτερο-Κωδικοποίηση PCM

2.1

2.2

3. Ερώτημα Τρίτο-Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-FSK

3.1

3.2

3.3

3.4

4.Παράρτημα με κώδικες

Ερώτημα Πρώτο-Κωδικοποίηση Huffman

1.1
Στο ερώτημα αυτό ζητείται η σύνταξη 3 συναρτήσεων οι οποίες να φέρνουν τα ίδια αποτελέσματα με τρεις ήδη υπάρχοντες συναρτήσεις της matlab (α) huffmandict (β) huffmanenco (γ) huffmandeco.

(α)Νέα υλοποίηση της huffmandict
Η huffmandict δεν έχει υλοποιηθεί ωστόσο για να μπορέσει να υλποιηθεί η υπόλοιπη άσκηση έχει χρησιμοποιηθεί η έτοιμη συνάρτηση της matlab.

(β)Νέα υλοποίηση της huffmanenco
Η συνάρτηση newhuffmanenco δέχεται ως είσοδο ένα σημα(inputSig) και ένα λεξικό(dict)το οποίο αντιστοιχεί κίθε σύμβολο του σήματος με την αντίστοιχη δυαδική του κωδικοποίηση αποτέλεσμα της συναρτησεις newhuffmandict. Δημιουργούνται δύο πίνακες, ο πίνακας n και ο πίνακας enco ,μέσα σε ένα for loop συγκρίνεται ένα ένα τα στοιχεία του σήματος εισόδου με την πρώτη στήλη του λεξικού και κρατιέται η θέση του.Έπειτα μέσα από μία ακόμη for loop στο πίνακα enco τοποθετείται η δυαδική αναπαράσταση του κάθε συμβόλου με την ίδια σειρά που εμφανίζεται στο inputSig.Έτσι, στο enco έχουμε την δυαδική κωδικοποιημένη μορφή του σήματος εισόδου(inputSig).

```
function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
 value=inputSig(i)
 ind = strcmp(value, dict)
 [row,col]=find(ind(:,1)==1)
 n=[n row ]
end
for i=1:length(n)
 enco=[enco dict(n(i),2)]
end
enco=cell2mat(enco)
end
```

(γ)Νέα υλοποίηση της huffmandenco
Η συνάρτηση huffmandenco παίρνει ως είσοδο το αποτέλεσμα της συνάρτησης huffmanenco και το λεξικό αποτέλεσμα της συνάρτηση newhuffmandict.Με την ίδια λογική αφού ο πίνακας enco έχει διαμορφωθεί στη σωστή μορφή συγκρίνεται κάθε στοιχείο του με την δεύτερη στήλη του λεξικό και κρατιέται η θέση του στουχείου.Έπειτα στο πίνακα sig τοποθετείται το σύμβολο που αντιστοιχεί στη κάθε δυαδική αναπαράσταση.Έτσι ως αποτέλεσμα έχουμε το σήμα που δόθηκε σε κωδικοποιημένη μορφή.

```
function [sig]=newhuffmandeco(enco,dict)
enco=num2cell(enco)
enco=string(enco)
dict=string(dict)
n=[]
sig = [];
for i=1:length(enco)
 value=enco(i)
 ind=strcmp(value,dict)
 [row,col]=find(ind==1)
 n=[n row ]
```

```
end
for i=1:length(n)
 sig=[sig dict(n(i),1)]
end
end
```

1.2

(α)

Στο ερώτημα αυτό αρχικά οφείλουμε να υπολογίσουμε τις πιθανότητες των συμβόλων της πηγής Α. Αρχικά,αποθηκεύεται η πηγή στη μεταβλητή s και έπειτα δημιουργείται ένας πίνακας που περιέχει την αγγλική αλφάβητο αλλά και το σύμβολο της δίεσης # το οποίο έχει αντικαταστίσει τα κενά.Μέσω μιας for loop υπολογίζεται η πιθανότητα κάθε συμβόλου διαιρώντας τις φορές εμφάνισεις με το μήκος του s χωρίς τα κενά.

```
s=importdata('cvxopt.txt')
str = cell2mat(s)
az='a':'z'
long=sum(ismember(str,az));
for k=1:numel(az)
  freq(k,1)=sum(ismember(str,az(k)))/long
end
```

Έπειτα καλούνται οι συναρτήσεις του 1.1 για να γίνει η κωδικοποίηση και η αποκωδικοποίηση για έλεγχο.

```
%Huffmandict
a = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't
    ' 'u' 'v' 'w' 'x' 'y' 'z' '#' }
p=freq.'
[dict] = huffmandict(a,p)

%Hufmanenco
[inputSig]=regexp(str, '\S', 'match')
[enco]=newhuffmanenco(inputSig,dict)

%Hufmandeco
[sig]=newhuffmandeco(enco,dict)
```

(β)

Η εντροπία υπολογίζεται από την πιθανότητα κάθε συμβόλου (p) με το log2(1/p).Στη περίπτωση μας το αποτέλεσμα είναι **4.1347**.Η τιμή αυτή είναι λογική καθώς η εντροπία μας παίρνει τιμές από το 0 εως το log2(27).

```
ent = -sum(p.*log2(p))
```

Το μέσο μήκος υπολογίζεται με το πολλαπλασιασμό κάθε πιθανότητας συμβόλου με το αντίστοιχο δυαδικό μήκος της huffman κωδικοποίησης.Στη περίπτωση μας το αποτέλεσμα είναι **4.1630**.

```
v=[]
for i=1:length(dict)
v=[v length(cell2mat(dict(i,2)))]
end
averagelength=sum(p.*v)
```

Η αποδοτικότητα δίνεται απο την εντροπία δια του μέσου μήκους.Στη περίπτωση μας το αποτέλεσμα είναι **99.3202**.

```
ef=100*(ent/averagelength)
```

1.3
Εδώ οι πιθανότητες μας δίνονται οπότε αρχικοποιούμε το λεξιλόγιο και της πιθανότητες κατευθείαν.Οι πιθανότητες που δίνονται δεν έχουν άθροισμα 1 οπότε αφαιρείται το απαραίτητο ώστε να είναι υλοποιήσιμη η κωδικοποίηση.

```
a = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't
    ' 'u' 'v' 'w' 'x' 'y' 'z' '#'}
p = [  0.0698 0.0128 0.0238 0.0364 0.1086 0.0190 0.0172 0.0521 0.0595 0.0013 0.0066
     0.0344 0.0206 0.0577 0.0642  0.0165 0.0008 0.0512 0.0541 0.0774 0.0236 0.0084
    0.0202 0.0010 0.0169 0.0006 0.1453]
s=importdata('cvxopt.txt')
str = cell2mat(s)
str = replace(str, ' ', '#')

%Huffmandict
[dict] = huffmandict(a,p)

%Huffmanenco
[inputSig]=regexp(str, '\S', 'match')
[enco]=newhuffmanenco(inputSig,dict)
```

Το μέσο μήκος υπολογίζεται από το ίδιο τύπο που έχει χρησημοποιηθεί στο ερώτημα 1.2.
Εδώ το μέσο μήκος είναι **4.1845** .Παρατηρείται μεγαλύτερο μέσο μήκος λογικό καθώς δεν έχει υπολογιστεί από την είσοδο.

1.4
Αρχικά στο κώδικα του ερωτήματος αυτού δημιουργείται το λεξιλόγιο της δεύτερης τάξης επέκταση της πηγής A δηλαδή κάθε πιθανή δυάδα από κάθε γράμμα του αγγλικού αλφαβήτου.Έπειτα υπολογίζεται η πιθανότητα εμφάνισεις κάθε δυάδας στη πηγή A.

```
az='a':'z'
[d]=regexp(az, '\S', 'match')

x={}
count=1
for i=1:length(d)
 for j=1:length(d)
  x(count)=strcat(d(i),d(j))
  count=count+1
  end
end

s=importdata('cvxopt.txt')
str= cell2mat(s)
str=str(find(~isspace(str)))
a=cellstr(reshape(str,2,[])')

long=sum(ismember(a,x))
for k=1:numel(x)
  freq(k,1)=sum(ismember(a,x(k)))/long
end
```

Το μέσο μήκος και η αποδοτικότητα του κώδικα υπολογίζονται από το ίδιο τύπο που έχει χρησημοποιηθεί στο ερώτημα 1.2.
Εδώ το μέσο μήκος είναι **7.4662** ενώ η αποδοτικότητα είναι **99.5937**.Παρατηρείται πολυ μεγαλύτερο μέσο μήκος το οποίο είναι πολύ λογικό καθώς έχουμε 26*26 σύμβολα, παρ'όλα αυτά η αποδοτικότητα έχει αυξηθεί.

1.5

Αρχικά, καλούνται οι συναρτήσεις του ερωτήματος 1.1 ,υπολογίζονται οι πιθανότητες και κωδικοποιείται η πηγή B.

```
 %Data
load('cameraman.mat')
b=unique(i)

x=reshape(i,[],1)

%Freq
long=sum(ismember(x,b))
for k=1:numel(b)
  freq(k,1)=sum(ismember(x,b(k)))/long
end

prob=freq.'

%Huffmandict
[dict]=huffmandict(b,prob)

%Huffmanenco
v=i(:).'
[enco]=newhuffmanenco(v,dict)
```

Το αποτέλεσμα της κωδικοποίησης εισάγεται στο bsc και παίρνουμε ως έξοδο το y.

```
%BSC
y=bsc(enco)
```

Για να υπολογίσουμε το p μετράμε πόσες θέσεις της εισόδου είναι ίσες με αυτές της εξόδου και διαιρούμε με το μήκος της εισόδου(ίδιο μήκος με έξοδο).Αυτή είναι η πιθανότητα σωστού εμείς θέλουμε πιθανότητα λάθους οπότε αφαιρούμε το ένα και κρατάμε ακρίβεια δύο ψηφίων.Έτσι, το p παίρνει την τιμή **0.18**

```
y=bsc(enco)

%P
count=0
for i=1:length(enco)
  if y(i)==enco(i)
      count=count+1
  end

end
p=1-count/length(enco)

p=round(p,2)
```

Η χωρητικότητα του καναλιού υπολογίζεται ως **0.3199** με τον παρακάτω τρόπο.

```
 %Capacity
H=-p*log2(p)-(1-p)*log2(1-p)
C=1-H
```

Η αμοιβαία πληροφορία υπολογίζεται ως **0.3208** με τον παρακάτω τρόπο.

```
 %Mutual information
J = [sum(~enco & ~y),sum(~enco & y);sum(enco & ~y),sum(enco& y)]/length(enco)
MI = sum(sum(J.*log2(J./(sum(J,2)*sum(J,1)))))

```

Ερώτημα Δεύτερο-Κωδικοποίηση PCM

Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

Η υλοποίηση του γίνεται με το αλγόριθμο Lloyd_Max. Αρχικά υπολογίζονται ο αριθμός των επίπεδων της κβάντισης και αφού υπολογίσουμε το εύρος κάθε περιοχής κβάντισης φτιάχνεται ο centers με τα κεντρα.0 αλγόριθμος υλοποιείται μέσω μιας while η οποία ισχύει όσο $|D_i - D_{(i-1)}| <$ε (στη αρχή η διαφορά αυτή ορίζεται ως 1).Έπειτα αρχικοποιείται το T όπως ορίζεται από την εκφώνηση της άσκησης και αφού έχει φτιαχτεί κβαντίζεται το σήμα. Τέλος υπολογίζεται η τρέχουσα παραμόρφωση και τα νέα κέντρα.

```
function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)

qlevels=2.^N
centers=zeros(qlevels,1)
T = zeros(qlevels+1,1)
step = (max_value+abs(min_value))/(qlevels)

centers = min_value + step/2 : step :max_value-(step/2)

 Dp=0
diff=1
count=1
epsilon = 10.^-6


while(diff>epsilon)

   T(1) = -inf
    for j=2:qlevels
        T(j)= 0.5*( centers(j-1)+centers(j) )
    end
    T(qlevels+1) = inf



    for i=1:length(x)
        for k=1:qlevels
           if ((x(i)<=T(k+1)) && (x(i)>T(k)))
                   xq(i)=centers(k)
           end
        end
    end


    d = mean((x-xq').^2)
    D(count) = d


    diff = abs(D(count)-Dp)
    Dp = D(count)

    for k=1:qlevels
        centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
    end


    count = count + 1
    end
 end
```

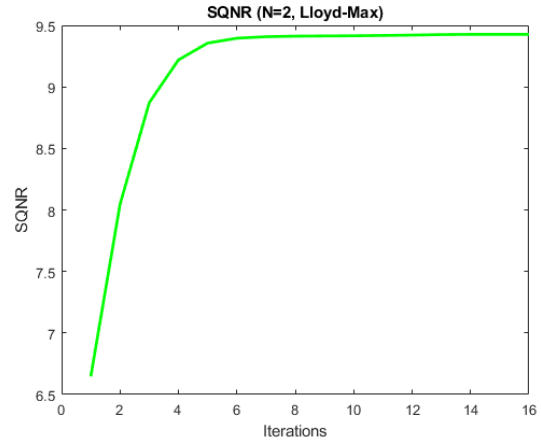
Μη Ομοιόμορφος Διανυσματικός Κβαντιστής

Καλείται η kmeans για το σήμα εισόδου x έπειτα για να κβαντιστεί και να φτιαχτεί το xq δημιουργείται μια for που για αντικαθιστά το x με το κέντρο του cluster στο οποίο ανήκει.

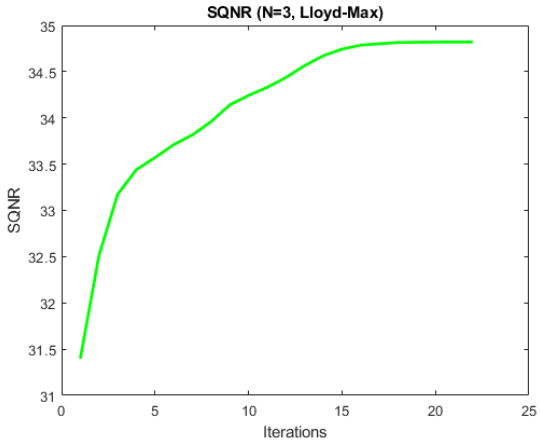
```
function [xq,centers,D]=vector_quant(x,k)

[idx,C] = kmeans(x,k)
centers=C

xq=[]
for i=1:length(idx)

  xq(i)=C(idx(i))
end
D = mean((x-xq)).^2
MEAND=mean(D)
end
```

2.1 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

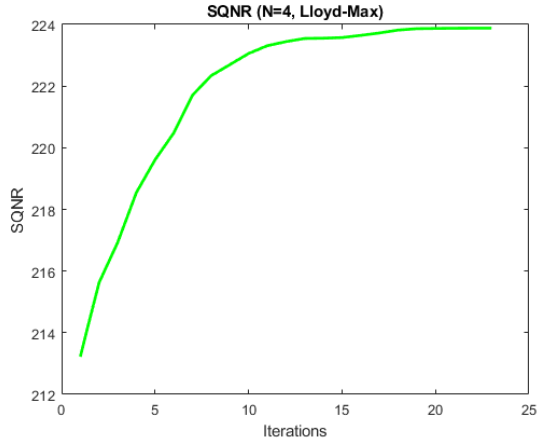
Για N=2 :Η μέση παραμόρφωση είναι 0.1380.



Για N=3: Η μέση παραμόρφωση είναι 0.0313.



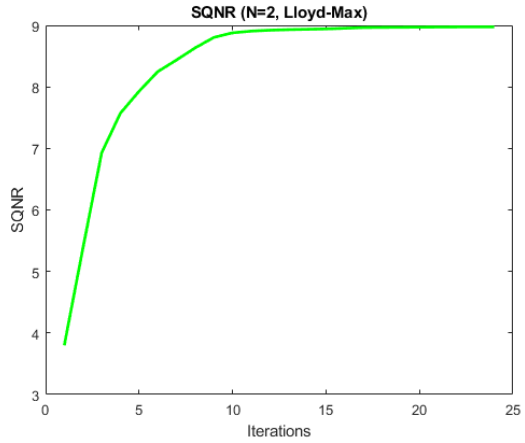
Για N=4:Η μέση παραμόρφωση είναι 0.0093.



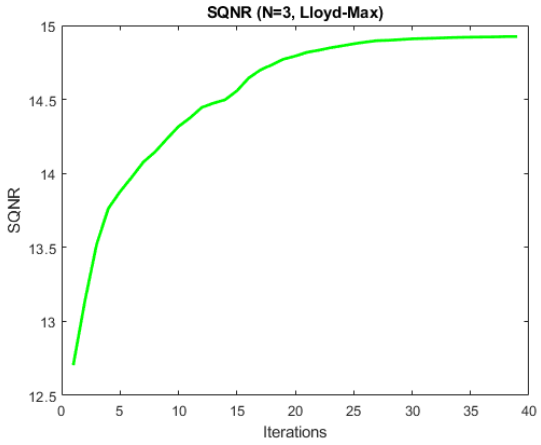
Ο διανυσματικός κβαντιστής δεν βγάζει σωστά αποτελέσματα.

2.2 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

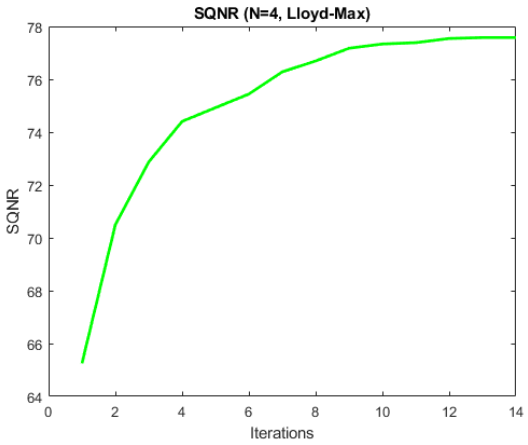
Για N=2 :Η μέση παραμόρφωση είναι = 0.1596.



Για N=3:Η μέση παραμόρφωση είναι 0.0472.



Για N=4:Η μέση παραμόρφωση είναι 0.0128.



Ο διανυσματικός κβαντιστής δεν βγάζει σωστά αποτελέσματα.

Ερώτημα Τρίτο-Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-FSK

3.1

Πομπός
Αρχικά δημιουργείται η δυαδική ακολουθία ίσων πιθανοτήτων 1 και 0 με την χρήση της συνάρτησης randn.Ο mapper που ακολουθεί μετατρέπει την δυαδική ακολουθία σε δεκαδική περνώντας ανάλογα με το M ,log2(M) ψηφιά της δυαδικής ακολουθίας. Έπειτα , δημιουργούνται τα M σήματα με τον τρόπο που δίνεται στη εκφώνηση μέσω μιας for και γίνεται το γινόμενο ανάλογα με τα σύμβολα που έχουν σταλθεί από τον mapper. Έτσι ,με την randn φτιάχνεται ο θόρυβος και προστίθεται στο προηγούμενο σήμα δίνοντας το σήμα εξόδου του πομπού.

Δέκτης
Ο δέκτης παίρνει ως είσοδο την έξοδο του πομπού. Το σήμα που δέχεται πολλαπλασιάζεται με τα M σήματα που είχαν φτιαχτεί από τον πομπό και αθροίζονται για κάθε μια θέση έτσι δημιουργείται το h που εμπεριέχει τα M r που θα εισαχθούν στον φωρατή. Στον φωρατή για κάθε M-αδα κρατιέται η θέση του μεγαλύτερου στοιχείου στο διάνυσμα f . Τέλος το f μετατρέπετε στη αντίστοιχη δυαδική ακολουθία οπού κάθε δεκαδικός αριθμός αντιστοιχείται σε δυαδικό με log2(M) ψηφιά ο καθένας.

Παρακάτω βρίσκεται ο κώδικας για κάθε κομμάτι

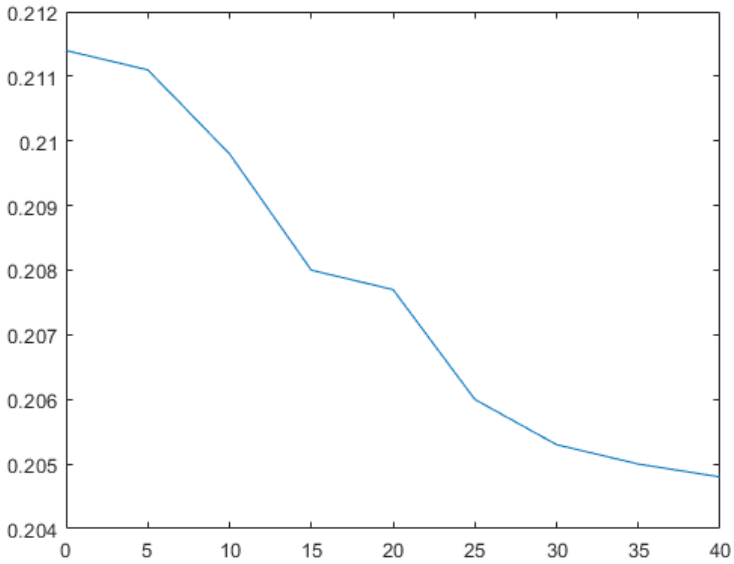
```
%Pompos
%Binary sequence
M=2
Lb=300000
bs = randsrc(Lb, 1, [0,1])


%Mapper
temp = mod(length(bs), log2(M))
n = bs(1 : (length(bs) - temp), :)
rn = reshape(n, log2(M), (length(bs) - temp) / log2(M))

symbols = bin2dec(num2str(rn'))


%Pulse
Tsymbol=4*10.^(-6)
Es = 1
g = sqrt(2 * Es / Tsymbol)


%FSK signals

fc=2.5*10.^6
Tsample=0.1*10.^(-6)
Tc=1/fc

sm = zeros(M, Tc/(Tsample))

t=Tc/(Tc/Tsample):Tc/(Tc/Tsample):Tc
for i = 1: M
  sm(i,:) =g*cos(2*pi*t*(fc+(i-1)/Tsymbol))
end

```

```
%Symbols to be sent

Syms=zeros(size(symbols,2),(Tc/Tsample)*Tsymbol/Tc)
Syms=repmat(sm(symbols+1,:),1,Tsymbol/Tc)

%AWGN

SNR=40
Eb = Es / log2(M)
N0 = Eb / (10^(SNR/10))
mean = 0
sigma = sqrt(N0 / 2)
noise = mean + sigma * randn(Lb/log2(M),1*40)


%R

rs = Syms + noise


%Dektis
%Fsk signals dektis
p=rs
h=zeros(Lb/log2(M),4)
rm=repmat(sm,1,Tsymbol/Tc)
h=mtimes(p,rm.')



%Foratis
[megisto,I] = max(h,[],2)


%Demapper
fn=I-1
np= dec2bin(fn,log2(M))
[lines, columns] = size(np)
ni = reshape(np', lines*columns, 1)
output= double(ni) - 48
```

3.2

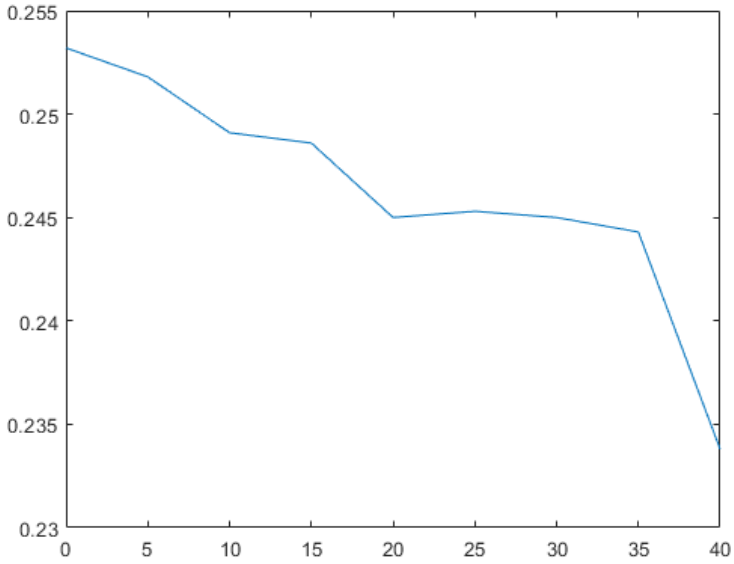
Η χρήση κωδικοποίησης Gray δεν έχει νόημα στο M-FSK .Αυτό συμβαίνει καθώς το σήμα που μεταδίδεται έχει ίση πιθανότητα να μπερδευτεί για κάποιο άλλο σήμα. Συνεπώς, τα πλεονεκτήματα της κωδικοποίηση Gray δεν αξιοποιούνται καθιστώντας την ανούσια στη παρούσα περίπτωση.

3.3
Ο υπολογισμός των καμπυλών BER έγινε με Lb=300000
Για M=8 η γραφική που προκύπτει είναι η παρακάτω



Δυστηχώς για τα υπόλοιπα M ο κώδικας για τις διάφορες τιμές του SNR βγάζει BER=0.

3.4
Ο υπολογισμός των καμπυλών SER έγινε με Lb=300000
Για M=8 η γραφική που προκύπτει είναι η παρακάτω



Δυστηχώς για τα υπόλοιπα M ο κώδικας για τις διάφορες τιμές του SNR βγάζει SER=0.

Παράρτημα με κώδικες

Ερώτημα Πρώτο-Κωδικοποίηση Huffman

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%1.1
function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
value=inputSig(i)
ind = strcmp(value, dict)
[row,col]=find(ind(:,1)==1)
n=[n row ]
end
for i=1:length(n)
enco=[enco dict(n(i),2)]

end
enco=cell2mat(enco)
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sig]=newhuffmandeco(enco,dict)
enco=num2cell(enco)
enco=string(enco)
dict=string(dict)
n=[]
sig = [];
for i=1:length(enco)
value=enco(i)
ind=strcmp(value,dict)
[row,col]=find(ind==1)
n=[n row ]
end
for i=1:length(n)

    sig=[sig dict(n(i),1)]

end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%1.2

%                    A
s=importdata('cvxopt.txt')
str = cell2mat(s)
str = replace(str, ' ', '#')
a = ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't
    ' 'u' 'v' 'w' 'x' 'y' 'z' '#' ]
long=sum(ismember(str,a));
for k=1:numel(a)
  freq(k,1)=sum(ismember(str,a(k)))/long
end


%Huffmandict
a = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't
    ' 'u' 'v' 'w' 'x' 'y' 'z' '#' }
```

```
p=freq.'
[dict] = huffmandict(a,p)

%Hufmanenco
[inputSig]=regexp(str, '\S', 'match')
[enco]=newhuffmanenco(inputSig,dict)


%Hufmandeco
[sig]=newhuffmandeco(enco,dict)

%Entropy
ent = -sum(p.*log2(p))

%Average length
v=[]
for i=1:length(dict)
v=[v length(cell2mat(dict(i,2)))]
end

averagelength=sum(p.*v)

%Efficiency of code
ef=100*(ent/averagelength)


%Functions

function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
value=inputSig(i)
ind = strcmp(value, dict)
[row,col]=find(ind(:,1)==1)
n=[n row ]
end
for i=1:length(n)
enco=[enco dict(n(i),2)]

end
enco=cell2mat(enco)
end


function [sig]=newhuffmandeco(enco,dict)
enco=num2cell(enco)
enco=string(enco)
dict=string(dict)
n=[]
sig = [];
for i=1:length(enco)
value=enco(i)
ind=strcmp(value,dict)
[row,col]=find(ind==1)
n=[n row ]
end
for i=1:length(n)

    sig=[sig dict(n(i),1)]

end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%1.3
%Data
a = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't
    ' 'u' 'v' 'w' 'x' 'y' 'z' '#'}
p = [  0.0698 0.0128 0.0238 0.0364 0.1086 0.0190 0.0172 0.0521 0.0595 0.0013 0.0066
     0.0344 0.0206 0.0577 0.0642  0.0165 0.0008 0.0512 0.0541 0.0774 0.0236 0.0084
    0.0202 0.0010 0.0169 0.0006 0.1453]
s=importdata('cvxopt.txt')
str = cell2mat(s)
str = replace(str, ' ', '#')

%Huffmandict
[dict] = huffmandict(a,p)

%Huffmanenco
[inputSig]=regexp(str, '\S', 'match')
[enco]=newhuffmanenco(inputSig,dict)

%Entropy
ent = -sum(p.*log2(p))

%Average length
v=[]
for i=1:length(dict)
v=[v length(cell2mat(dict(i,2)))]
end

averagelength=sum(p.*v)


%Functions

function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
value=inputSig(i)
ind = strcmp(value, dict)
[row,col]=find(ind(:,1)==1)
n=[n row ]
end
for i=1:length(n)
enco=[enco dict(n(i),2)]

end
enco=cell2mat(enco)
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1.4
az='a':'z'
[d]=regexp(az, '\S', 'match')

x={}
count=1
for i=1:length(d)
```

```
 for j=1:length(d)
  x(count)=strcat(d(i),d(j))
  count=count+1
  end
end


s=importdata('cvxopt.txt')
str= cell2mat(s)
str=str(find(~isspace(str)))
a=cellstr(reshape(str,2,[])')

long=sum(ismember(a,x))
for k=1:numel(x)
  freq(k,1)=sum(ismember(a,x(k)))/long
end

[dict]=huffmandict(x,freq)

[enco]=newhuffmanenco(a,dict)


p=freq.'

%Entropy
pnz=nonzeros(p)
ent = -sum(pnz.*log2(pnz))

%Average length
v=[]
for i=1:length(dict)
v=[v length(cell2mat(dict(i,2)))]
end

averagelength=sum(p.*v)

%Efficiency of code
ef=100*(ent/averagelength)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
value=inputSig(i)
ind = strcmp(value, dict)
[row,col]=find(ind(:,1)==1)
n=[n row ]
end
for i=1:length(n)
enco=[enco dict(n(i),2)]

end
enco=cell2mat(enco)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%1.5

```

```
%Data
load('cameraman.mat')
b=unique(i)

x=reshape(i,[],1)

%Freq
long=sum(ismember(x,b))
for k=1:numel(b)
  freq(k,1)=sum(ismember(x,b(k)))/long
end

prob=freq.'

%Huffmandict

[dict]=huffmandict(b,prob)

%Huffmanenco
v=i(:).'

[enco]=newhuffmanenco(v,dict)

%BSC

y=bsc(enco)

%P
count=0
for i=1:length(enco)
  if y(i)==enco(i)
      count=count+1
  end

end
p=1-count/length(enco)

p=round(p,2)

%Capacity
H=-p*log2(p)-(1-p)*log2(1-p)

C=1-H

%Mutual info
J = [sum(~enco & ~y),sum(~enco & y);sum(enco & ~y),sum(enco& y)]/length(enco)
MI = sum(sum(J.*log2(J./(sum(J,2)*sum(J,1)))))


%Functions
function [enco]=newhuffmanenco(inputSig,dict)
n=[]
enco = [];
for i=1:length(inputSig)
value=inputSig(i)
ind = strcmp(value, dict)
[row,col]=find(ind(:,1)==1)
n=[n row ]
end
for i=1:length(n)
enco=[enco dict(n(i),2)]

```

```
end
enco=cell2mat(enco)
end
```

Ερώτημα Δεύτερο-Κωδικοποίηση PCM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)


qlevels=2.^N
centers=zeros(qlevels,1)
T = zeros(qlevels+1,1)
step = (max_value+abs(min_value))/(qlevels)


centers = min_value + step/2 : step :max_value-(step/2)

Dp=0
diff=1
count=1
epsilon = 10.^-6


while(diff>epsilon)



    T(1) = -inf
    for j=2:qlevels
        T(j)= 0.5*( centers(j-1)+centers(j) )
    end
    T(qlevels+1) = inf



    for i=1:length(x)
        for k=1:qlevels
           if ((x(i)<=T(k+1)) && (x(i)>T(k)))
                   xq(i)=centers(k)
           end
        end
    end


    d = mean((x-xq').^2)
    D(count) = d


    diff = abs(D(count)-Dp)
    Dp = D(count)



   for k=1:qlevels
        centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
    end


    count = count + 1
    end
 end

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xq,centers,D]=vector_quant(x,k)

[idx,C] = kmeans(x,k)
centers=C

xq=[]
for i=1:length(idx)

  xq(i)=C(idx(i))
end
D = mean((x-xq)).^2
MEAND=mean(D)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%2.1
%Π   A
M=10000
x= randn(M,1)

%B
%N=2
[xq2,centers2,D2] = Lloyd_Max(x,2,max(x),min(x))
SQNR2 = mean(x.^2)./D2
err2=mean(mean((x-xq2).^2))
%N=3
[xq3,centers3,D3] = Lloyd_Max(x,3,max(x),min(x))
SQNR3 = mean(x.^2)./D3
err3 =mean(mean((x-xq3).^2))

%N=4
[xq4,centers4,D4] = Lloyd_Max(x,4,max(x),min(x))
SQNR4 = mean(x.^2)./D4
err4 =mean(mean((x-xq4).^2))


%Δ

%N=2 k=4
%[xqv2,centers,Dv2]=vector_quant(x,4)
%SQNRv2 = mean(x.^2)./Dv2
%errv2 = =mean(mean((x-xqv2).^2))

%N=3 k=6
%[xqv4,centers,Dv3]=vector_quant(x,6)
%SQNRv3 = mean(x.^2)./Dv3
%errv3 = mean(mean((x-xqv3).^2))
%N=4 k=8
%[xqv4,centers,Dv4]=vector_quant(x, 8)
%SQNRv4 = mean(x.^2)./Dv4
%errv4 =mean(mean((x-xqv4).^2))

%Σ    mean distrortion
%N=2
Md2=mean(D2)-mean(Dv2)
%N=3
Md3=mean(D3)-mean(Dv3)
%N=4
Md4=mean(D4)-mean(Dv4)


```

```
function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)


qlevels=2.^N
centers=zeros(qlevels,1)
T = zeros(qlevels+1,1)
step = (max_value+abs(min_value))/(qlevels)


centers = min_value + step/2 : step :max_value-(step/2)



Dp=0
diff=1
count=1
epsilon = 1e-7


while(diff>e)



    T(1) = -inf
    for j=2:qlevels
        T(j)= 0.5*( centers(j-1)+centers(j) )
    end
    T(qlevels+1) = inf



    for i=1:length(x)
        for k=1:qlevels
           if ((x(i)<=T(k+1)) && (x(i)>T(k)))
                   xq(i)=centers(k)
           end
        end
    end


    d = mean((x-xq').^2)
    D(count) = d


    diff = abs(D(count)-Dp)
    Dp = D(count)



   for k=1:qlevels
        centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
    end


    count = count + 1

end

end

function [xq,centers,D]=vector_quant(x,k)

```

```
[idx,C] = kmeans(x,k)
centers=C

xq=[]
for i=1:length(idx)

  xq(i)=C(idx(i))
end
D = (x-xq).^2
MEAND=mean(D)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%2.2
%Π  B
M=1000
x = randn(M,1)
b = 1
a = [1 1/2 1/3 1/4 1/5 1/6 ]
y = filter(b,a,x)

%B
%N=2
[xq2,centers2,D2] = Lloyd_Max(y,2,max(y),min(y))
SQNR2 = mean(y.^2)./D2
err2 =mean(mean((x-xq2).^2))

%N=3
[xq3,centers3,D3] = Lloyd_Max(y,3,max(y),min(y))
SQNR3 = mean(y.^2)./D3
err3 =mean(mean((x-xq3).^2))

%N=4
[xq4,centers4,D4] = Lloyd_Max(y,4,max(y),min(y))
SQNR4 = mean(y.^2)./D4
err4 = mean(mean((x-xq4).^2))

%N=2 k=4
%[xqv2,centers,Dv2]=vector_quant(y,4)
%SQNRv2 = mean(y.^2)./Dv2
%errv2 = immse(y-xqv2)

%N=3 k=6
%[xqv4,centers,Dv3]=vector_quant(y,6)
%SQNRv3 = mean(y.^2)./Dv3
%errv3 = immse(y-xqv3)

%N=4 k=8
%[xqv4,centers,Dv4]=vector_quant(y, 8)
%SQNRv4 = mean(y.^2)./Dv4
%errv4 = immse(y-xqv4)




function [xq,centers,D]=vector_quant(x,k)

[idx,C] = kmeans(x,k)
centers=C

xq=[]
for i=1:length(idx)
```

```
244
245   xq(i)=C(idx(i))
246 end
247 D = (x-xq).^2
248 MEAND=mean(D)
249 end
```

Ερώτημα Τρίτο-Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-FSK

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %Π
3  %Δ
4  M=8
5  Lb=300000
6  bs = randsrc(Lb, 1, [0,1])
7
8  %Mapper
9  temp = mod(length(bs), log2(M))
10 n = bs(1 : (length(bs) - temp), :)
11 rn = reshape(n, log2(M), (length(bs) - temp) / log2(M))
12
13 symbols = bin2dec(num2str(rn'))
14
15 %Pulse
16 Tsymbol=4*10.^(-6)
17 Es = 1
18 g = sqrt(2 * Es / Tsymbol)
19
20
21 %FSK signals
22
23 fc=2.5*10.^6
24 Tsample=0.1*10.^(-6)
25 Tc=1/fc
26
27 sm = zeros(M, Tc/(Tsample))
28
29 t=Tc/(Tc/Tsample):Tc/(Tc/Tsample):Tc
30 for i = 1: M
31   sm(i,:) =g*cos(2*pi*t*(fc+(i-1)/Tsymbol))
32 end
33
34
35 %Symbols to be sent
36 Syms=zeros(size(symbols,2),(Tc/Tsample)*Tsymbol/Tc)
37
38 Syms=repmat(sm(symbols+1,:),1,Tsymbol/Tc)
39
40 %AWGN
41
42 SNR=40
43 Eb = Es / log2(M)
44 N0 = Eb / (10^(SNR/10))
45 mean = 0
46 sigma = sqrt(N0 / 2)
47 noise = mean + sigma * randn(Lb/log2(M),1*40)
48
49 %R
50 rs = Syms + noise
51
52 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53 %Δ
54 %Fsk signals
```

```
55
56 p=rs
57 h=zeros(Lb/log2(M),4)
58 rm=repmat(sm,1,Tsymbol/Tc)
59 h=mtimes(p,rm.')
60
61 %Φ
62
63 [megisto,I] = max(h,[],2)
64
65 %Demapper
66
67 fn=I-1
68 np= dec2bin(fn,log2(M))
69 [lines, columns] = size(np)
70 ni = reshape(np', lines*columns, 1)
71 output= double(ni) - 48
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73 BER = sum(bs ~= output)/length(output)
74
75 SER=sum(symbols ~= fn)/length(fn)
```