

Ψηφιακές Τηλεπικοινωνίες

Εργαστηριακή Άσκηση Σετ 1

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 υλοποιηθεί η υπόλοιπη άσκηση έχει χρησιμοποιηθεί η έτοιμη συνάρτηση της matlab.

(β)Νέα υλοποίηση της huffmanenco

Η συνάρτηση newhuffmanenco δέχεται ως είσοδο ένα σήμα(inputSig) και ένα λεξικό(dict)το οποίο αντιστοιχεί κάθε σύμβολο του σήματος με την αντίστοιχη δυαδική του κωδικοποίηση αποτέλεσμα της συναρτήσεων 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).

```
1 function [enco]=newhuffmanenco(inputSig,dict)
2 n=[]
3 enco = [];
4 for i=1:length(inputSig)
5     value=inputSig(i)
6     ind = strcmp(value, dict)
7     [row,col]=find(ind(:,1)==1)
8     n=[n row ]
9 end
10 for i=1:length(n)
11     enco=[enco dict(n(i),2)]
12 end
13 enco=cell2mat(enco)
14 end
```

(γ)Νέα υλοποίηση της huffmandenco

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

```
1 function [sig]=newhuffmandeco(enco,dict)
2 enco=num2cell(enco)
3 enco=string(enco)
4 dict=string(dict)
5 n=[]
6 sig = [];
7 for i=1:length(enco)
8     value=enco(i)
9     ind=strcmp(value,dict)
10    [row,col]=find(ind==1)
11    n=[n row ]
```

```

12 end
13 for i=1:length(n)
14     sig=[sig dict(n(i),1)]
15 end
16 end

```

1.2

(α)

Στο ερώτημα αυτό αρχικά οφείλουμε να υπολογίσουμε τις πιθανότητες των συμβόλων της πηγής A. 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 αντικαταστήσει τα κενά. Μέσω μιας for loop υπολογίζεται η πιθανότητα κάθε συμβόλου διαιρώντας τις φορές εμφάνισεις με το μήκος του s χωρίς τα κενά.

```

1 s=importdata('cvxopt.txt')
2 str = cell2mat(s)
3 az='a':'z'
4 long=sum(ismember(str,az));
5 for k=1:numel(az)
6     freq(k,1)=sum(ismember(str,az(k)))/long
7 end

```

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

```

1 %Huffmandict
2 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' '#'}
3 p=freq.'
4 [dict] = huffmandict(a,p)
5
6 %Huffmanenco
7 [inputSig]=regexp(str, '\S', 'match')
8 [enco]=newhuffmanenco(inputSig,dict)
9
10 %Hufmandeco
11 [sig]=newhufmandeco(enco,dict)

```

(β)

Η εντροπία υπολογίζεται από την πιθανότητα κάθε συμβόλου (p) με το $\log_2(1/p)$. Στη περίπτωση μας το αποτέλεσμα είναι **4.1347**. Η τιμή αυτή είναι λογική καθώς η εντροπία μας παίρνει τιμές από το 0 έως το $\log_2(27)$.

```

1 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**.

```

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

```

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

```

1 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.

```
1 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' '#'}  
2 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]  
3 s=importdata('cvxopt.txt')  
4 str = cell2mat(s)  
5 str = replace(str, ' ', '#')  
6  
7 %Huffmandict  
8 [dict] = huffmandict(a,p)  
9  
10 %Huffmanenco  
11 [inputSig]=regexp(str, '\S', 'match')  
12 [enco]=newhuffmanenco(inputSig,dict)
```

Το μέσο μήκος υπολογίζεται από το ίδιο τύπο που έχει χρησιμοποιηθεί 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 εμφάνισης κάθε δυάδας στη πηγή A.

```
1 az='a':'z'  
2 [d]=regexp(az, '\S', 'match')  
3  
4 x={}  
5 count=1  
6 for i=1:length(d)  
7     for j=1:length(d)  
8         x(count)=strcat(d(i),d(j))  
9         count=count+1  
10    end  
11 end  
12  
13 s=importdata('cvxopt.txt')  
14 str= cell2mat(s)  
15 str=str(find(~isspace(str)))  
16 a=cellstr(reshape(str,2,[]))  
17  
18 long=sum(ismember(a,x))  
19 for k=1: numel(x)  
20     freq(k,1)=sum(ismember(a,x(k)))/long  
21 end
```

Το μέσο μήκος και η αποδοτικότητα του κώδικα υπολογίζονται από το ίδιο τύπο που έχει χρησιμοποιηθεί στο ερώτημα 1.2.

Εδώ το μέσο μήκος είναι **7.4662** ενώ η αποδοτικότητα είναι **99.5937**. Παρατηρείται πολύ 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.

```
1 %Data
2 load('cameraman.mat')
3 b=unique(i)
4
5 x=reshape(i,[],1)
6
7 %Freq
8 long=sum(ismember(x,b))
9 for k=1:numel(b)
10     freq(k,1)=sum(ismember(x,b(k)))/long
11 end
12
13 prob=freq.'
14
15 %Huffmandict
16 [dict]=huffmandict(b,prob)
17
18 %Huffmanenco
19 v=i(:).''
20 [enco]=newhuffmanenco(v,dict)
```

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

```
1 %BSC
2 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**

```
1 y=bsc(enco)
2
3 %P
4 count=0
5 for i=1:length(enco)
6     if y(i)==enco(i)
7         count=count+1
8     end
9
10 end
11 p=1-count/length(enco)
12
13 p=round(p,2)
```

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

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

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

```
1 %Mutual information
2 J = [sum(~enco & ~y),sum(~enco & y);sum(enco & ~y),sum(enco & y)]/length(enco)
3 MI = sum(sum(J.*log2(J./(sum(J,2)*sum(J,1))))
4
```

Ερώτημα Δεύτερο-Κωδικοποίηση PCM

Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

Η υλοποίηση του γίνεται με το αλγόριθμο Lloyd_Max. Αρχικά υπολογίζονται ο αριθμός των επιπέδων της κβάντισης και αφού υπολογίσουμε το εύρος κάθε περιοχής κβάντισης φτιάχνεται ο centers με τα κέντρα. Ο αλγόριθμος υλοποιείται μέσω μιας `while` η οποία ισχύει όσο $|D_i - D_{(i-1)}| < \epsilon$ (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.

```
1 function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)
2
3 qllevels=2.^N
4 centers=zeros(qllevels,1)
5 T = zeros(qllevels+1,1)
6 step = (max_value+abs(min_value))/(qllevels)
7
8 centers = min_value + step/2 : step :max_value-(step/2)
9
10 Dp=0
11 diff=1
12 count=1
13 epsilon = 10.^-6
14
15
16 while(diff>epsilon)
17
18     T(1) = -inf
19     for j=2:qllevels
20         T(j)= 0.5*( centers(j-1)+centers(j) )
21     end
22     T(qllevels+1) = inf
23
24
25
26     for i=1:length(x)
27         for k=1:qllevels
28             if ((x(i)<=T(k+1)) && (x(i)>T(k)))
29                 xq(i)=centers(k)
30             end
31         end
32     end
33
34
35     d = mean((x-xq').^2)
36     D(count) = d
37
38
39     diff = abs(D(count)-Dp)
40     Dp = D(count)
41
42     for k=1:qllevels
43         centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
44     end
45
46
47     count = count + 1
48 end
49 end
```

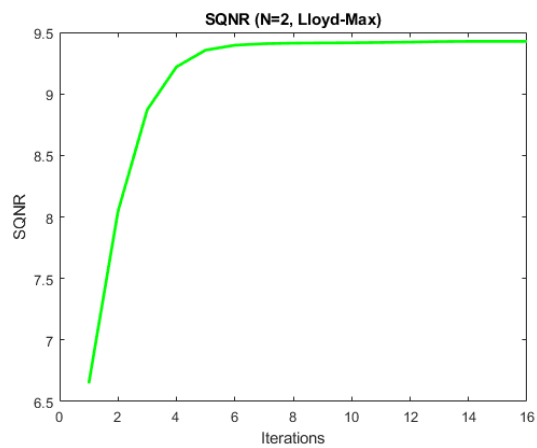
Μη Ομοιόμορφος Διανυσματικός Κβαντιστής

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.

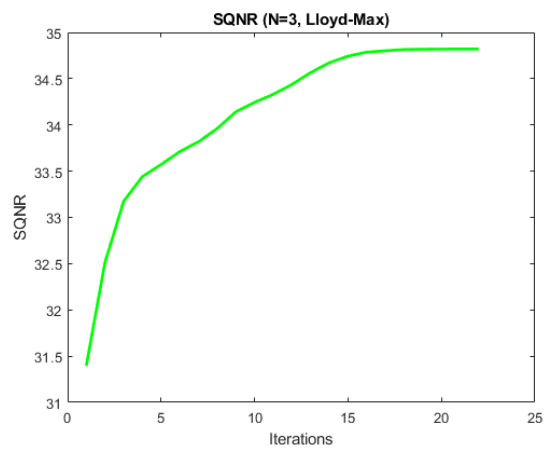
```
1 function [xq,centers,D]=vector_quant(x,k)
2
3 [idx,C] = kmeans(x,k)
4 centers=C
5
6 xq=[]
7 for i=1:length(idx)
8
9     xq(i)=C(idx(i))
10 end
11 D = mean((x-xq)).^2
12 MEAND=mean(D)
13 end
```


2.1 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

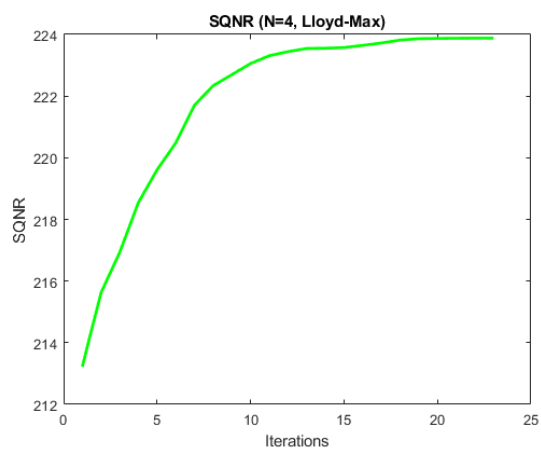
Για $N=2$: Η μέση παραμόρφωση είναι 0.1380.



Για $N=3$: Η μέση παραμόρφωση είναι 0.0313.



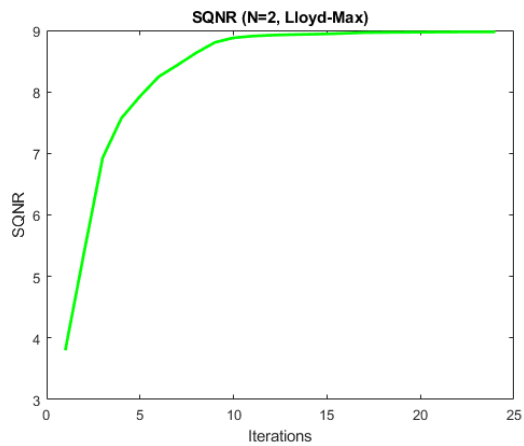
Για $N=4$: Η μέση παραμόρφωση είναι 0.0093.



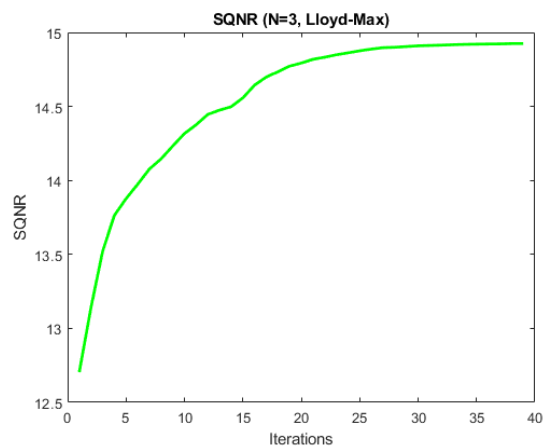
Ο διανυσματικός κβαντιστής δεν βγάζει σωστά αποτελέσματα.

2.2 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

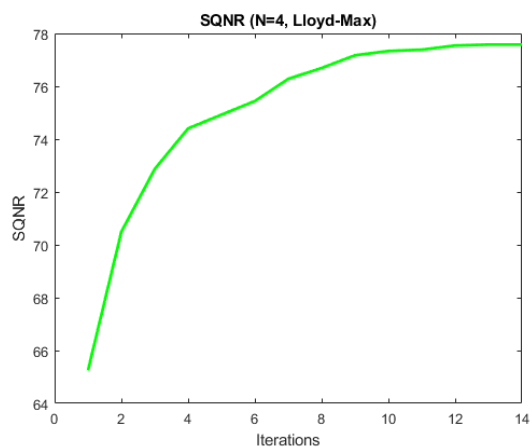
Για $N=2$:Η μέση παραμόρφωση είναι $= 0.1596$.



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 , $\log_2(M)$ ψηφία 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 μετατρέπεται στη αντίστοιχη δυαδική ακολουθία όπου κάθε δεκαδικός αριθμός αντιστοιχείται σε δυαδικό με $\log_2(M)$ ψηφία ο καθένας.

Παρακάτω βρίσκεται ο κώδικας για κάθε κομμάτι

```
1 %Pompos
2 %Binary sequence
3 M=2
4 Lb=300000
5 bs = randsrc(Lb, 1, [0,1])
6
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
16 %Pulse
17 Tsymbol=4*10.^(-6)
18 Es = 1
19 g = sqrt(2 * Es / Tsymbol)
20
21
22 %FSK signals
23
24 fc=2.5*10.^6
25 Tsample=0.1*10.^(-6)
26 Tc=1/fc
27
28 sm = zeros(M, Tc/(Tsample))
29
30 t=Tc/(Tc/Tsample):Tc/(Tc/Tsample):Tc
31 for i = 1: M
32     sm(i,:) =g*cos(2*pi*t*(fc+(i-1)/Tsymbol))
33 end
34
```

```

35 %Symbols to be sent
36
37 Syms=zeros(size(symbols,2),(Tc/Tsample)*Tsymbol/Tc)
38 Syms=repmat(sm(symbols+1,:),1,Tsymbol/Tc)
39
40 %AWGN
41
42 SNR=40
43 Eb = Es / log2(M)
44 NO = Eb / (10^(SNR/10))
45 mean = 0
46 sigma = sqrt(NO / 2)
47 noise = mean + sigma * randn(Lb/log2(M),1*40)
48
49
50 %R
51
52 rs = Syms + noise
53
54
55 %Dektis
56 %Fsk signals dektis
57 p=rs
58 h=zeros(Lb/log2(M),4)
59 rm=repmat(sm,1,Tsymbol/Tc)
60 h=mtimes(p,rm.')
61
62
63
64 %Foratis
65 [megisto,I] = max(h,[],2)
66
67
68 %Demapper
69 fn=I-1
70 np= dec2bin(fn,log2(M))
71 [lines, columns] = size(np)
72 ni = reshape(np', lines*columns, 1)
73 output= double(ni) - 48

```

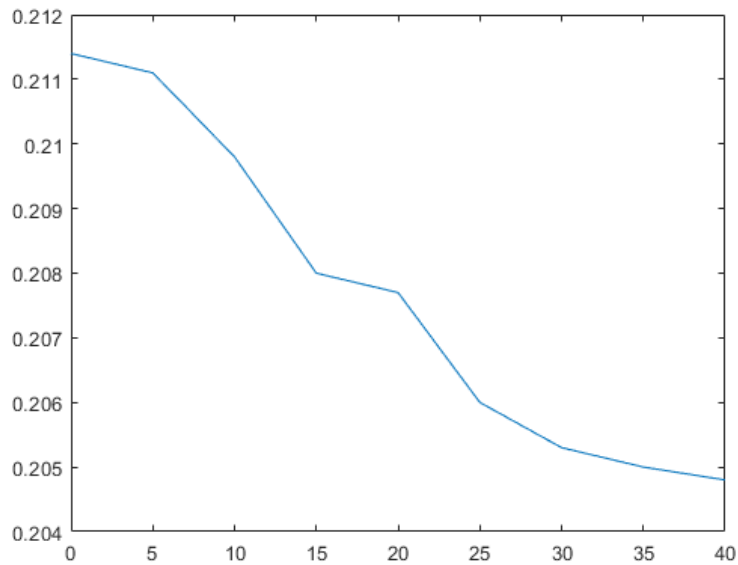
3.2

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 κωδικοποίησης Gray δεν αξιοποιούνται καθιστώντας την ανούσια στη παρούσα περίπτωση.

3.3

Ο υπολογισμός των καμπυλών BER έγινε με $L_b=300000$

Για $M=8$ η γραφική που προκύπτει είναι η παρακάτω

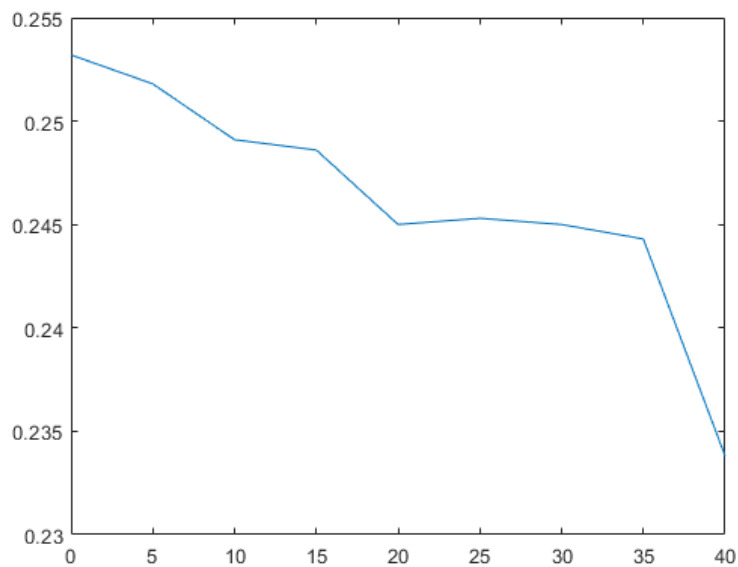


Δυστηγώς για τα υπόλοιπα M ο κώδικας για τις διάφορες τιμές του SNR βγάζει $BER=0$.

3.4

Ο υπολογισμός των καμπυλών SER έγινε με $L_b=300000$

Για $M=8$ η γραφική που προκύπτει είναι η παρακάτω



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

Παράρτημα με κώδικες

Ερώτημα Πρώτο-Κωδικοποίηση Huffman

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %1.1
3 function [enco]=newhuffmanenco(inputSig,dict)
4 n=[]
5 enco = [];
6 for i=1:length(inputSig)
7 value=inputSig(i)
8 ind = strcmp(value, dict)
9 [row,col]=find(ind(:,1)==1)
10 n=[n row ]
11 end
12 for i=1:length(n)
13 enco=[enco dict(n(i),2)]
14
15 end
16 enco=cell2mat(enco)
17 end
18
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21
22 function [sig]=newhuffmandeco(enco,dict)
23 enco=num2cell(enco)
24 enco=string(enco)
25 dict=string(dict)
26 n=[]
27 sig = [];
28 for i=1:length(enco)
29 value=enco(i)
30 ind=strcmp(value,dict)
31 [row,col]=find(ind==1)
32 n=[n row ]
33 end
34 for i=1:length(n)
35
36     sig=[sig dict(n(i),1)]
37
38 end
39 end
40
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42 %1.2
43
44 %
45 %A
46 s=importdata('cvxopt.txt')
47 str = cell2mat(s)
48 str = replace(str, ' ', '#')
49 a = ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't'
50      'u' 'v' 'w' 'x' 'y' 'z' '#']
51 long=sum(ismember(str,a));
52 for k=1:numel(a)
53     freq(k,1)=sum(ismember(str,a(k)))/long
54 end
55
56 %Huffmandict
57 a = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't'
58      'u' 'v' 'w' 'x' 'y' 'z' '#' }
```

```

57 p=freq.'
58 [dict] = huffmandict(a,p)
59
60 %Huffmanenco
61 [inputSig]=regexp(str, '\S', 'match')
62 [enco]=newhuffmanenco(inputSig,dict)
63
64
65 %Huffmandeco
66 [sig]=newhuffmandeco(enco,dict)
67
68 %Entropy
69 ent = -sum(p.*log2(p))
70
71 %Average length
72 v=[]
73 for i=1:length(dict)
74 v=[v length(cell2mat(dict(i,2)))]
75 end
76
77 averagelength=sum(p.*v)
78
79 %Efficiency of code
80 ef=100*(ent/averagelength)
81
82
83 %Functions
84
85 function [enco]=newhuffmanenco(inputSig,dict)
86 n=[]
87 enco = [];
88 for i=1:length(inputSig)
89 value=inputSig(i)
90 ind = strcmp(value, dict)
91 [row,col]=find(ind(:,1)==1)
92 n=[n row ]
93 end
94 for i=1:length(n)
95 enco=[enco dict(n(i),2)]
96
97 end
98 enco=cell2mat(enco)
99 end
100
101
102 function [sig]=newhuffmandeco(enco,dict)
103 enco=num2cell(enco)
104 enco=string(enco)
105 dict=string(dict)
106 n=[]
107 sig = [];
108 for i=1:length(enco)
109 value=enco(i)
110 ind=strcmp(value,dict)
111 [row,col]=find(ind==1)
112 n=[n row ]
113 end
114 for i=1:length(n)
115
116     sig=[sig dict(n(i),1)]
117
118 end

```

```

119 end
120
121 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122 %1.3
123 %Data
124 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' '#'}
125 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]
126 s=importdata('cvxopt.txt')
127 str = cell2mat(s)
128 str = replace(str, ' ', '#')
129
130 %Huffmandict
131 [dict] = huffmandict(a,p)
132
133 %Huffmanenco
134 [inputSig]=regexp(str, '\S', 'match')
135 [enco]=newhuffmanenco(inputSig,dict)
136
137 %Entropy
138 ent = -sum(p.*log2(p))
139
140 %Average length
141 v=[]
142 for i=1:length(dict)
143 v=[v length(cell2mat(dict(i,2)))]
144 end
145
146 averagelength=sum(p.*v)
147
148
149 %Functions
150
151 function [enco]=newhuffmanenco(inputSig,dict)
152 n=[]
153 enco = [];
154 for i=1:length(inputSig)
155 value=inputSig(i)
156 ind = strcmp(value, dict)
157 [row,col]=find(ind(:,1)==1)
158 n=[n row ]
159 end
160 for i=1:length(n)
161 enco=[enco dict(n(i),2)]
162
163 end
164 enco=cell2mat(enco)
165 end
166
167
168
169 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
170
171 %1.4
172 az='a':'z'
173 [d]=regexp(az, '\S', 'match')
174
175 x={}
176 count=1
177 for i=1:length(d)

```



```

178 for j=1:length(d)
179     x(count)=strcat(d(i),d(j))
180     count=count+1
181 end
182 end
183
184
185 s=importdata('cvxopt.txt')
186 str= cell2mat(s)
187 str=str(find(~isspace(str)))
188 a=cellstr(reshape(str,2,[]))
189
190 long=sum(ismember(a,x))
191 for k=1: numel(x)
192     freq(k,1)=sum(ismember(a,x(k)))/long
193 end
194
195 [dict]=huffmandict(x,freq)
196
197 [enco]=newhuffmanenco(a,dict)
198
199
200 p=freq.'
201
202 %Entropy
203 pnz=nonzeros(p)
204 ent = -sum(pnz.*log2(pnz))
205
206 %Average length
207 v=[]
208 for i=1:length(dict)
209 v=[v length(cell2mat(dict(i,2)))]
210 end
211
212 averagelength=sum(p.*v)
213
214 %Efficiency of code
215 ef=100*(ent/averagelength)
216
217
218 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
219
220
221 function [enco]=newhuffmanenco(inputSig,dict)
222 n=[]
223 enco = [];
224 for i=1:length(inputSig)
225 value=inputSig(i)
226 ind = strcmp(value, dict)
227 [row,col]=find(ind(:,1)==1)
228 n=[n row ]
229 end
230 for i=1:length(n)
231 enco=[enco dict(n(i),2)]
232
233 end
234 enco=cell2mat(enco)
235 end
236
237 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
238 %1.5
239

```

```

240 %Data
241 load('cameraman.mat')
242 b=unique(i)
243
244 x=reshape(i,[],1)
245
246 %Freq
247 long=sum(ismember(x,b))
248 for k=1:numel(b)
249     freq(k,1)=sum(ismember(x,b(k)))/long
250 end
251
252 prob=freq.'
253
254 %Huffmandict
255
256 [dict]=huffmandict(b,prob)
257
258 %Huffmanenco
259 v=i(:). '
260
261 [enco]=newhuffmanenco(v,dict)
262
263 %BSC
264
265 y=bsc(enco)
266
267 %P
268 count=0
269 for i=1:length(enco)
270     if y(i)==enco(i)
271         count=count+1
272     end
273
274 end
275 p=1-count/length(enco)
276
277 p=round(p,2)
278
279 %Capacity
280 H=-p*log2(p)-(1-p)*log2(1-p)
281
282 C=1-H
283
284 %Mutual info
285 J = [sum(~enco & ~y),sum(~enco & y);sum(enco & ~y),sum(enco& y)]/length(enco)
286 MI = sum(sum(J.*log2(J./(sum(J,2)*sum(J,1))))))
287
288
289 %Functions
290 function [enco]=newhuffmanenco(inputSig,dict)
291 n=[]
292 enco = [];
293 for i=1:length(inputSig)
294     value=inputSig(i)
295     ind = strcmp(value, dict)
296     [row,col]=find(ind(:,1)==1)
297     n=[n row ]
298 end
299 for i=1:length(n)
300     enco=[enco dict(n(i),2)]
301

```

```

302 end
303 enco=cell2mat(enco)
304 end

```

Ερώτημα Δεύτερο-Κωδικοποίηση PCM

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)
3
4
5 qlevels=2.^N
6 centers=zeros(qlevels,1)
7 T = zeros(qlevels+1,1)
8 step = (max_value+abs(min_value))/(qlevels)
9
10
11 centers = min_value + step/2 : step :max_value-(step/2)
12
13 Dp=0
14 diff=1
15 count=1
16 epsilon = 10.^-6
17
18
19 while(diff>epsilon)
20
21
22
23     T(1) = -inf
24     for j=2:qlevels
25         T(j)= 0.5*( centers(j-1)+centers(j) )
26     end
27     T(qlevels+1) = inf
28
29
30
31     for i=1:length(x)
32         for k=1:qlevels
33             if ((x(i)<=T(k+1)) && (x(i)>T(k)))
34                 xq(i)=centers(k)
35             end
36         end
37     end
38
39
40     d = mean((x-xq').^2)
41     D(count) = d
42
43
44     diff = abs(D(count)-Dp)
45     Dp = D(count)
46
47
48
49     for k=1:qlevels
50         centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
51     end
52
53
54     count = count + 1
55     end
56 end
57

```

```

58 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59
60 function [xq,centers,D]=vector_quant(x,k)
61
62 [idx,C] = kmeans(x,k)
63 centers=C
64
65 xq=[]
66 for i=1:length(idx)
67     xq(i)=C(idx(i))
68 end
69 D = mean((x-xq).^2)
70 MEAND=mean(D)
71 end
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73 %2.1
74 %Π A
75 M=10000
76 x= randn(M,1)
77
78 %B
79 %N=2
80 [xq2,centers2,D2] = Lloyd_Max(x,2,max(x),min(x))
81 SQNR2 = mean(x.^2)./D2
82 err2=mean(mean((x-xq2).^2))
83 %N=3
84 [xq3,centers3,D3] = Lloyd_Max(x,3,max(x),min(x))
85 SQNR3 = mean(x.^2)./D3
86 err3 =mean(mean((x-xq3).^2))
87
88 %N=4
89 [xq4,centers4,D4] = Lloyd_Max(x,4,max(x),min(x))
90 SQNR4 = mean(x.^2)./D4
91 err4 =mean(mean((x-xq4).^2))
92
93
94 %Δ
95
96 %N=2 k=4
97 [xqv2,centers,Dv2]=vector_quant(x,4)
98 %SQNRv2 = mean(x.^2)./Dv2
99 %errv2 = mean(mean((x-xqv2).^2))
100
101 %N=3 k=6
102 [xqv4,centers,Dv3]=vector_quant(x,6)
103 %SQNRv3 = mean(x.^2)./Dv3
104 %errv3 = mean(mean((x-xqv3).^2))
105
106 %N=4 k=8
107 [xqv4,centers,Dv4]=vector_quant(x,8)
108 %SQNRv4 = mean(x.^2)./Dv4
109 %errv4 =mean(mean((x-xqv4).^2))
110
111 %Σ mean distrortion
112 %N=2
113 Md2=mean(D2)-mean(Dv2)
114 %N=3
115 Md3=mean(D3)-mean(Dv3)
116 %N=4
117 Md4=mean(D4)-mean(Dv4)
118
119

```

```

120 function [xq,centers,D] = Lloyd_Max(x,N,max_value,min_value)
121
122
123 qllevels=2.^N
124 centers=zeros(qllevels,1)
125 T = zeros(qllevels+1,1)
126 step = (max_value+abs(min_value))/(qllevels)
127
128
129 centers = min_value + step/2 : step :max_value-(step/2)
130
131
132
133 Dp=0
134 diff=1
135 count=1
136 epsilon = 1e-7
137
138
139 while(diff>e)
140
141
142
143     T(1) = -inf
144     for j=2:qllevels
145         T(j)= 0.5*( centers(j-1)+centers(j) )
146     end
147     T(qllevels+1) = inf
148
149
150
151     for i=1:length(x)
152         for k=1:qllevels
153             if ((x(i)<=T(k+1)) && (x(i)>T(k)))
154                 xq(i)=centers(k)
155             end
156         end
157     end
158
159
160     d = mean((x-xq').^2)
161     D(count) = d
162
163
164     diff = abs(D(count)-Dp)
165     Dp = D(count)
166
167
168
169     for k=1:qllevels
170         centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ))
171     end
172
173
174     count = count + 1
175
176 end
177
178 end
179
180 function [xq,centers,D]=vector_quant(x,k)
181

```

```

182 [idx,C] = kmeans(x,k)
183 centers=C
184
185 xq=[]
186 for i=1:length(idx)
187
188     xq(i)=C(idx(i))
189 end
190 D = (x-xq).^2
191 MEAND=mean(D)
192 end
193
194 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
195 %2.2
196 %Π B
197 M=1000
198 x = randn(M,1)
199 b = 1
200 a = [1 1/2 1/3 1/4 1/5 1/6 ]
201 y = filter(b,a,x)
202
203 %B
204 %N=2
205 [xq2,centers2,D2] = Lloyd_Max(y,2,max(y),min(y))
206 SQNR2 = mean(y.^2)./D2
207 err2 =mean(mean((x-xq2).^2))
208
209 %N=3
210 [xq3,centers3,D3] = Lloyd_Max(y,3,max(y),min(y))
211 SQNR3 = mean(y.^2)./D3
212 err3 =mean(mean((x-xq3).^2))
213
214 %N=4
215 [xq4,centers4,D4] = Lloyd_Max(y,4,max(y),min(y))
216 SQNR4 = mean(y.^2)./D4
217 err4 = mean(mean((x-xq4).^2))
218
219 %N=2 k=4
220 [xqv2,centers,Dv2]=vector_quant(y,4)
221 %SQNRv2 = mean(y.^2)./Dv2
222 %errv2 = immse(y-xqv2)
223
224 %N=3 k=6
225 [xqv4,centers,Dv3]=vector_quant(y,6)
226 %SQNRv3 = mean(y.^2)./Dv3
227 %errv3 = immse(y-xqv3)
228
229 %N=4 k=8
230 [xqv4,centers,Dv4]=vector_quant(y, 8)
231 %SQNRv4 = mean(y.^2)./Dv4
232 %errv4 = immse(y-xqv4)
233
234
235
236
237 function [xq,centers,D]=vector_quant(x,k)
238
239 [idx,C] = kmeans(x,k)
240 centers=C
241
242 xq=[]
243 for i=1:length(idx)

```

```

244     xq(i)=C(idx(i))
245
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 NO = Eb / (10^(SNR/10))
45 mean = 0
46 sigma = sqrt(NO / 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 %Phi
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)

```