Πρώτο σετ εργαστηριακών ασκήσεων

Κωνσταντοπούλου Ευαγγελία

22 Ιανουαρίου 2021

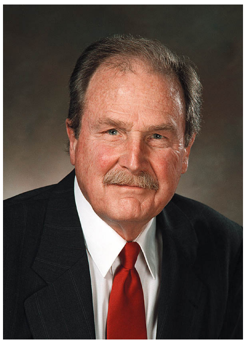
Ψηφιακές τηλεπικοινωνίες

Περιεχόμενα

Για τη διευκόλυνσή σας, έχω παρεμβάλει ορισμένες αριθμημένες γραμμές κώδικα, ώστε η εξήγηση της υλοποίησης μου να είναι κατανοητή. Όλος ο κώδικας σε MATLAB όμως, βρίσκεται στο τέλος της αναφοράς όπως ζητήσατε.

1 Ερώτημα 1 - Κωδικοποίηση Huffman

Γενικά - Ιστορία

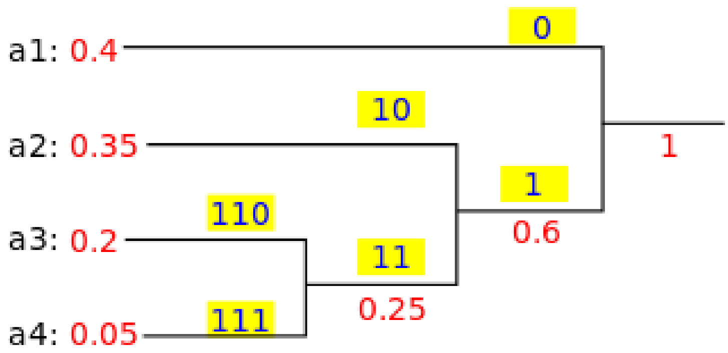


Η κωδικοποίηση Huffman είναι μια μέθοδος συμπίεσης που δημοσιεύτηκε το 1952 από τον David Huffman. Παράγει ένα κώδικα βασισμένο στην πιθανότητα εμφάνισης του κάθε συμβόλου σε ένα κείμενο. Σχεδόν σε όλα τα κείμενα, μερικά σύμβολα εμφανίζονται περισσότερες φορές από ότι άλλα. Προκαθορισμένες πιθανότητες εμφάνισης κάθε συμβόλου χρησιμοποιούνται για τη δημιουργία ενός πλήρους δυαδικού δέντρου από τη βάση προς τα επάνω (bottom-up). Αυτός ο τρόπος εγγυάται ότι τα σύμβολα που εμφανίζονται λιγότερο θα έχουν μακρύτερες σειρές δυαδικών ψηφίων. Στο δέντρο τα σύμβολα είναι φύλλα (τερματικοί κόμβοι - terminal nodes), οι διακλαδώσεις σημειώνονται με 0 ή 1 και η δυαδική αναπαράσταση της διαδρομής από τη ρίζα (root) μέχρι το σύμβολο είναι η συμπιεσμένη αναπαράστασή του ως σειρά δυαδικών ψηφίων.

Βήματα του αλγορίθμου

Δημιουργία Δυαδικού Δέντρου:

1. Διάταξε τις εισόδους κατά φθίνουσα σειρά πιθανοτήτων
2. Συγχώνευσε τα δύο σύμβολα με τις μικρότερες πιθανότητες και δημιούργησε νέο «σύμβολο»
3. Ανάθεσε στα δύο συνιστώντα σύμβολα «0» και «1»
4. Ταξινόμησε εκ νέου τη λίστα των συμβόλων
5. Επανέλαβε τα παραπάνω μέχρις όλα τα σύμβολα συγχωνευτούν σε ένα τελικό σύμβολο



Σχήμα 1: Παράδειγμα κωδικοποίησης Huffman για σύμβολα a1, a2, a3, a4

Στο δυαδικό δέντρο που δημιουργήθηκε:

- ρίζα: το τελικό σύνθετο σύμβολο
- φύλλα: τα αρχικά σύμβολα
- ενδιάμεσοι κόμβοι: σύνθετα σύμβολα

Ανάθεση Bits σε Σύμβολα Εισόδου:

1. Ξεκίνα από τη ρίζα και κινήσου προς ένα φύλλο
2. Η ακολουθία των bits που συναντώνται είναι η ακολουθία κωδικοποίησης
3. Επανέλαβε για όλα τα σύμβολα (φύλλα)

Μειονεκτήματα:

- Είναι απαραίτητη η εκ των προτέρων γνώση των πιθανοτήτων εμφάνισης κάθε συμβόλου
- Δεν προσφέρεται για εφαρμογές πραγματικού χρόνου

Πλεονέκτημα:

- Επιτυγχάνει το ελάχιστο μέσο μήκος κώδικα ανάμεσα σε όλους τους symbol-by-symbol προθεματικούς κώδικες, όταν έχουμε symbol-by-symbol coding και γνωστή pdf.

1.1 Ερώτημα 1:huffmandict, huffmanenco και huffmandeco

1.1.1 huffmandict

Παρακάτω παρατίθεται η υλοποίηση κώδικα συναρτήσεων ανάλογης της huffmandict που δίνεται από τη MATLAB. Η συνάρτηση newhuffmandict δέχεται ως ορίσματα ένα cell array συμβόλων s και των αντίστοιχων πιθανοτήτων τους σε ένα διάνυσμα p, και επιστρέφει ένα cell array-κωδικοποίηση Huffman ως δυαδική συμβολοσειρά που ονομάζω finalcode. Κάθε λέξη του finalcode αντιστοιχεί σε ένα σύμβολο του οποίου η πιθανότητα βρίσκεται στον ανάλογο δείκτη του p.

Αρχικά, πρέπει να σιγουρευτώ πως οι παράμετροι που εισάγαμε είναι της μορφής που ζητάμε. Για το p, ελέγχω ότι έχει δύο διαστάσεις, ότι περιέχει τουλάχιστον μια τιμή, ότι οι τιμές του είναι πραγματικοί αριθμοί και ότι είναι αριθμητικός (γραμμή 3 του κώδικα). Εάν κάτι από όσα προαναφέραμε δεν ισχύει, τυπώνεται το αντίστοιχο μήνυμα λάθους. Για το αλφάβητο συμβόλων s δε, ελέγχω εάν είναι της μορφής cell array, και εάν το μήκος του συμφωνεί με αυτό του p(γραμμές 6-11). Διαφορετικά, ενημερώνουμε ανάλογα με μήνυμα λάθους.

Στην περίπτωση που το αλφάβητο εισόδου είναι ένα και μόνο σύμβολο, ορίζω απέυθείας ως αντίστοιχη κωδική λέξη το 1. Αλλιώς, καλώ με τη σειρά τις συναρτήσεις sorting με ορίσματα τα s και p, και traversetree με ορίσματα το s και την κωδική λέξη που έχει φτιαχτεί ως στιγμής, δηλαδή την κενή. Ας εξηγήσω τον ρόλο των συναρτήσεων αυτών.

sorting: Ο αλγόριθμος Huffman θα σταματήσει όταν έχουν μείνει δύο μόνο πιθανότητες/σύμβολα, επομένως όλα όσα εξηγήσω στη συνέχεια γίνονται μόνο εάν το πλήθος των συμβόλων στο τωρινό βήμα είναι μεγαλύτερο του 2. Ξεκινώ με τη διάταξη των πιθανοτήτων, το οποίο υλοποιώ με τη γραμμή 26. Παρατηρήστε ότι κρατάω μέσω του δείκτη idx τον τρόπο που άλλαξαν σειρά σε σχέση με την αρχική τους θέση στο διάνυσμα p. Αυτό το κάνω για να αναδιατάξω με τον ίδιο τρόπο τα σύμβολα s. Συμπτύσσω τις δύο μικρότερες πιθανότητες στις πρώτες δύο θέσεις του διανύσματος (γραμμή 27) και τοποθετώ το άθροισμά τους στη θέση της δεύτερης. Την πρώτη την κάνουμε κενή. Αντίστοιχα επεξεργάζομαι και τα δύο πρώτα κελιά του s. Αυτό είναι σημαντικό για την τελική ικανοποίηση του while loop.

traversetree: Ορίζουμε την τελική κωδική λέξη ως global ώστε να μην αλλοιώνεται από τις αναδρομικές κλήσεις της συνάρτησης traversetree. Θυμηθείτε ότι στην ουσία φτιάχνουμε ένα πλήρες δυαδικό δέντρο το οποίο πρέπει να διατρέξουμε ώστε να παράγουμε τις τελικές κωδικές λέξεις. Εάν εξετάζουμε έναν κόμβο του cell array, πρέπει να αναθέσουμε κατά τη διάτρεξη στα παιδιά του τις τιμές 0 ή 1. Κάνω την παράδοχη ότι το πρώτο παιδί θα παίρνει την τιμή 0 (γραμμή 39) και το δεύτερο την τιμή 1. Θα προστίθεται δηλαδή στο τέλος της λέξης το αντίστοιχο ψηφίο. Όταν φτασω στα φύλλα του δέντρου, τυπώνω την δημιουργηθείσα λέξη.

Ας δοκιμάσουμε τη συνάρτησή μας με ένα παράδειγμα. Έστω ότι εισάγω στο command window της MATLAB ως s το cell array s={1;2;3;4;5;6} και το διάνυσμα πιθανοτήτων p=[.1,.4,.06,.1,.04,.3]. Καλώ τη συνάρτηση με την εντολή call = newhuffmandict(s,p). Αυτό που θα μου τυπωθεί είναι το cell array finalcode με τις κωδικές λέξεις του προκύπτουν.

```
Command Window
>> s={1;2;3;4;5;6}

s =

  6×1 cell array

    {[1]}
    {[2]}
    {[3]}
    {[4]}
    {[5]}
    {[6]}

>> p=[.1,.4,.06,.1,.04,.3]

p =

    0.1000    0.4000    0.0600    0.1000    0.0400    0.3000

>> call = newhuffmandict(s,p)

call =

  6×2 cell array

    {[1]}    {'0111' }
    {[2]}    {'1'    }
    {[3]}    {'01101'}
    {[4]}    {'010'  }
    {[5]}    {'01100'}
fx  {[6]}    {'00'   }
```

Σχήμα 2: Δοκιμή συνάρτησης newhuffmandict

Παρατήρηση 1: Προκύπτει ένας προθεματικός κώδικας, δηλαδή καμία από τις λέξεις που θα προκύψουν δε θα είναι πρόθεμα μια άλλης λέξης.

Παρατήρηση 2: Περιμένουμε ότι τα σύμβολα που έχουν τις μικρότερες πιθανότητες εμφάνισης θα έχουν μακρύτερες σειρές δυαδικών ψηφίων. Όντως η μεγαλύτερη κωδική λέξη αντιστοιχεί στο σύμβολο 5 με τη μικρότερη πιθανότητα (0.06).

Κώδικας:

```
function finalcode = newhuffmandict(s,p)

if (ndims(p) ~= 2) | (min(size(p))>1) | ~isreal(p) | ~isnumeric(p)
error('O p prepei na einai arithmitiko dianusma pragmatikwn
    arithmwn');
end
if ~isa(s, 'cell')
    error('To alphabhto symbolwn prepei na einai ths morfhs cell
        array')
end
if length(p) ~= length(s)
    error('Prepei to plithos twn stoixeiwn toy dianusmatos
        pithanothtwn kai tou alphabhtou symbolwn na symfwnoun')
end

global finalcode
finalcode = cell(length(p),2);
if length(p) > 1
%p = p/sum(p);
s = sorting(s,p);
traversetree(s, []);
else
finalcode = {'1'};
end

function s = sorting(s,p)

while numel(s) > 2
[p, idx] = sort(p);
p(2) = p(1) + p(2);
p(1) = [];
s = s(idx);
```

```
30 s{2} = {s{1},s{2}};
31 s(1) = [];
32 end
33
34
35 function traversetree(node, word)
36
37 global finalcode
38 if isa(node,'cell')
39 traversetree(node{1},[word 0]);
40 traversetree(node{2}, [word 1]);
41 else
42 finalcode{node,1} = node;
43 %finalcode{node,2} = char('0' + word);
44 finalcode{node,2} = word;
45 %each Huffman codeword is represented as a row vector
46 end
```

ΠΡΟΣΟΧΗ: Η δεύτερη στήλη του finalcode τυπώνεται ως χαρακτήρες μέσω της γραμμής 43 για να δείξω την ορθότητα του κώδικα. Όμως για τον έλεγχο της huffmanenco χρειάζεται να είναι της μορφής double, όπως είναι αρχικά δηλαδή. Επομένως θα αντικαταστήσω αυτή τη γραμμή με το: finalcodenode,2 = word;

1.1.2 huffmanenco

Παρακάτω παρατίθεται η υλοποίηση κώδικα συναρτήσεων ανάλογης της huffmanenco που δίνεται από τη MATLAB. Η συνάρτηση newhuffmanenco δέχεται ως ορίσματα το σήμα προς κωδικοποίηση shma, που είναι διάνυσμα είτε γραμμής είτε στήλης, και το λεξικό που παράχθηκε από τη συνάρτηση newhuffmandict. Το αποτέλεσμα θα είναι το κωδικοποιημένο σήμα μας, encoded.
Αρχικά κάνουμε κάποιους ελέγχους για τα arguments που εισάγουμε κατά την κλήση. Με τον έλεγχο της γραμμής 5 απαιτώ ότι το σήμα που εισάγω είναι μορφής διανύσματος. Αν ικανοποιείται αυτή η συνθήκη, παίρνω τις διαστάσεις του. Αυτές τις χρειάζομαι ώστε μέσω του mat2cell να διαμελίσω το διάνυσμα, τοποθετώντας κάθε στοιχείο μεμονωμένα σε ένα κελί ενός cell array που ονομάζω πάλι shma.

Συνεχίζοντας, πρέπει να δεσμεύσουμε αρκετή μνήμη για το διάνυσμα encoded που θα περιέχει το αποτέλεσμα της κωδικοποίησης. Το μέγιστο της μνήμης

```
Command Window
>> sig = [ 2 1 4 2 1 1 5 4 ]

sig =

     2     1     4     2     1     1     5     4

>> c= newhuffmanenco(sig,call)

c =

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

>> c1= huffmanenco(sig,call)

c1 =

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

fx >>
```

Σχήμα 3: Δοκιμή συνάρτησης newhuffmanenco

που θα χρειαζόμουν θα ήταν εάν όλα τα στοιχεία του σήματος αντιστοιχούσαν στη λέξη του λεξικού με το μεγαλύτερο μήκος. Βρίσκω ποια είναι αυτή μέσω ενός for loop (17) που διατρέχει τη δεύτερη στήλη του λεξικού και των μεταβλητών largeSz και tmp που θα κρατούν το την κωδική λέξη με το μέγιστο μήκος (20). Ύστερα δεσμεύω με μηδενικά το μέγιστο διάνυσμα που υπάρχει περίπτωση να χρειαστώ, δηλαδή εάν έχει μήκος όσο το μήκος του σήματος επί τη μέγιστη κωδική λέξη (24).

Ώρα να κάνουμε λοιπόν την κωδικοποίηση. Για κάθε τιμή του σήματος, θα πρέπει να την ψάξω στην πρώτη στήλη του λεξικού, ώστε να βρω σε ποια κωδική λεξη αντιστοιχεί. Χρησιμοποιώ ένα βοηθητικό διάνυσμα ithcode μήκους ithcodelgth στο οποίο βάζω την κωδική λέξη που βρήκα σε κάθε iteration του λεξικού. Κάθε φορά που τελειώνω την αντιστοίχιση για μια λέξη του σήματος (32), πρέπει να αντιγράψω το αποτέλεσμα στο τελικό διάνυσμα που θα επιστέψει η συνάρτηση, encoded. Μέσω του δείκτη index που δείχνει στο τέλος της προηγούμενης λέξης που γράφτηκε, ο οποίος θα μετακινηθεί στο τέλος της νέας λέξης μετά την εγγραφή (39), φροντίζει ώστε να γραφτούν με τη σειρά όλες οι λέξεις χωρίς επικάλυψη.

Επειδή μάλλον έχουμε αρχικοποιήσει μεγαλύτερο διάνυσμα απ' αυτό που γεμίσαμε τελικά, κρατάω το encoded μέχρι εκεί που δείχνει τελευταία φορά ο index (42). Έτσι, δε θα τυπωθούν παραπλανητικά μηδενικά που δεν αντιστοιχούν στην κωδικοποίηση κάποιας λέξης. Τέλος, επειδή είπαμε πως η είσοδος μπορεί να είναι μορφής διανύσματος στήλης, με έλεγχο αν η διάσταση n είναι μονάδα, φροντίζω να τυπωθεί το encoded κάθετα, για διευκόλυνσή μας.

Ας τρέξουμε τη συνάρτησή μας με ένα παράδειγμα. Έστω s,p τα ίδια του προηγούμενου ερωτήματος και call το λεξικό που δημιουργήθηκε από την κλήση της newhuffmandict. Θεωρώ ως σήμα εισόδου το διάνυσμα sig=[2 1 4 2 1 1 5 4]. Τρέχοντας τη newhuffmanenco με ορίσματα το sig και call, και την in-built συνάρτηση huffmanenco της MATLAB παίρνω ακριβώς τα ίδια αποτελέσματα.

Κώδικας:

```
function encoded = newhuffmanenco(shma, dict)

[m,n] = size(shma);

if (m==1 || n==1)
  [m,n] = size(shma);
  shma = mat2cell(shma, ones(1,m), ones(1,n) );
end


largeSz = 0;
dictSz = size(dict,1);

for i = 1 : dictSz
    tmp = size(dict{i,2},2);
    if (tmp > largeSz)
        largeSz = tmp;
    end
end

encoded = zeros(1, length(shma)*largeSz);

index = 1;
for i = 1 : length(shma)

    ithcode = [];
    for j = 1 : dictSz
        if( shma{i} == dict{j,1} )
            ithcode = dict{j,2};
            break;
        end
    end
```

```

    ithcodelgth = length(ithcode);
    encoded(index : index + ithcodelgth - 1) = ithcode;
    index = index + ithcodelgth;
end

encoded = encoded(1:index-1);

if( n == 1 )
    encoded = encoded';
end
```

1.1.3 huffmandeco

Παρακάτω παρατίθεται η προσπάθεια υλοποίησης συνάρτησης ανάλογης της huffmandeco. Δυστυχώς, παρότι δεν εμφανίζεται κάποιο error κατά το debugging, δεν τυπώνεται το ορθό αποτέλεσμα. Παρ΄όλα αυτά θα εξηγήσω τη σκέψη πίσω από την προσέγγισή μου. Ξεκινάμε περνώντας στην μεταβλητή size_of_dic το μέγεθος του λεξικού ώστε να μπορέσω να το διατρέξω (3). Το symbol είναι το αλφαριθμητικό με τα σύμβολα που θα ανατήσω απ΄το λεξικό προς εκτύπωση, το οποίο αρχικοποιώ ως κενό.

Διατρέχω λοιπόν το λεξικό με μια for loop (9). Το διάνυσμα dictEnco θα περιέχει την κωδικοποιημένη λέξη που διάβασε στη δεύτερη στήλη του εισαγόμενου λεξικού dict. Υπολογίζω το μήκος της λέξης αυτής και το συγκρίνω με τη λέξη kwdikas που έχω διαβάσει μέχρι στιγμής από το encoded σήμα. Αυτό αποτελεί παράμετρο εισόδου για τη συνάρτηση find η οποία θα καλεστεί αργότερα. Εάν το μήκος των δύο λέξεων δεν είναι συμβατό, προχωράω στην επόμενη λέξη του λεξικού. Εάν έχουν το ίδιο μήκος, κάνω έλεγχο εάν οι δύο λέξεις είναι ίδιες. Τότε, ορίζω ως symbol, το σύμβολο της πρώτης στήλης του λεξικού, που βρίσκεται στην ίδια γραμμή j με την κωδικοποιημένη dictEnco.

Βρήκα λοιπόν πώς θα ψάχνω για τη σωστή λέξη. Ώρα να βρούμε πως θα γίνεται η ανάγνωση κωδικών λέξεων από το encoded, και η εκτύπωση των συμβόλων στο τέλος της συνάρτησης. Ορίζω ως n το μήκος του εισαγόμενου διανύσματος encoded (μπορεί να είναι είτε διάνυσμα γραμμής είτε στήλης). Η μεταβλητή start δείχνει στην αρχή της λέξης του διανύσματος encoded που εξετάζουμε, ενώ η finish στο τέλος αυτής. Αρχικά, καθώς θέλουμε να εξετάσουμε το πρώτο ψηφίο του διανύσματος, ορίζω ως start=finish=1. Η ακόλουθη διαδικασία που θα περιγράψω, επαναλαμβάνεται μέχρις ότου η μεταβλητή finish να πάρει την τιμή n, δηλαδή μέχρι να φτάσουμε στο τέλος του encoded.

Καλώ τη συνάρτηση find που είδαμε προηγουμένως, με όρισμα kwdikas= encoded(start:finish) (34). Το επεστραμένο σύμβολο symbol θα το τοποθετήσω στο τέλος του διανύσματος decoded το οποίο στο τέλος θα περιέχει το αποκωδικοποιημένο σήμα (36). Για να διαβάσω την επόμενη δυαδική λέξη προς αποκωδικοποίηση, ορίζω ως δείκτη start, τον προηγούμενο δείκτη finish + 1 και τον finish τον μεταφέρω δεξιά τόσες θέσεις, όσες το μήκος της dictEncoz που κατάφερα να αποκωδικοποιήσω.

Κώδικας:

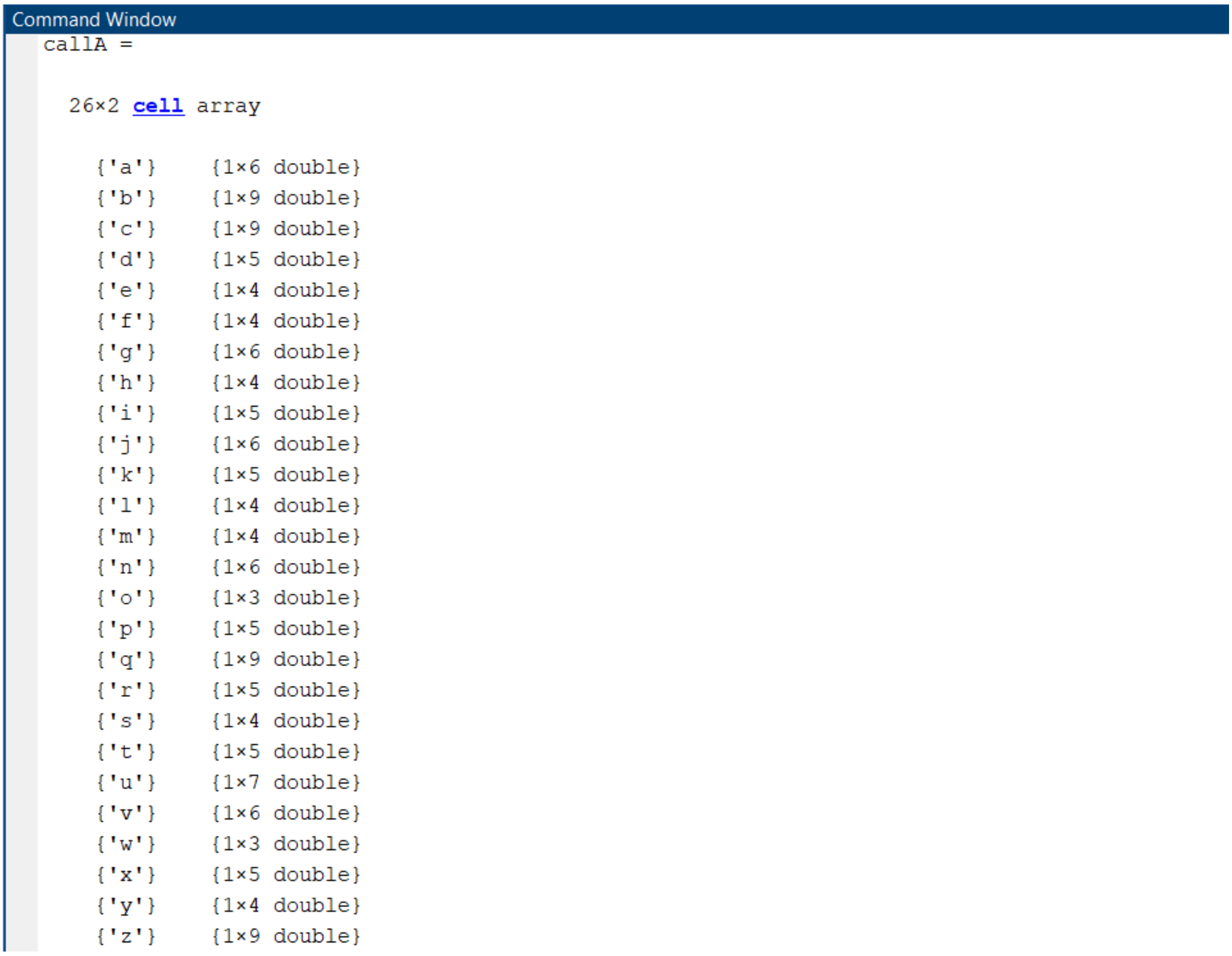
```
function [decoded] = stackdeco(encoded,dict)

size_of_dic = size(dict,1);

function [dictEncosz,symbol] = find(kwdikas)
symbol=[];


for j = 1:size_of_dic
    dictEnco = cell2mat(dict(j,2));
    dictEncosz = size(dictEnco, 2);
   size = size(kwdikas, 2);
    if dictEncosz > size
        break;
    end
    if isequal(size,dictEncosz) && isequal(dictEnco,kwdikas)
        symbol = cell2mat(dict(j,1));
        break;
    end
end
end

start = 1;
finish = 1;
decoded = [];
[n1, n2] = size(encoded);
if n1 > n2
    n = n1;
else
    n = n2;
end

while finish < n
   [dictEncosz,symbol] = find(encoded(start:finish));
```

```

    decoded =[decoded symbol];

    start = finish + 1;
    finish = finish + dictEncosz;

end
```

1.2 Ερώτημα 2

1.2.1 Κωδικοποίηση Πηγής Α

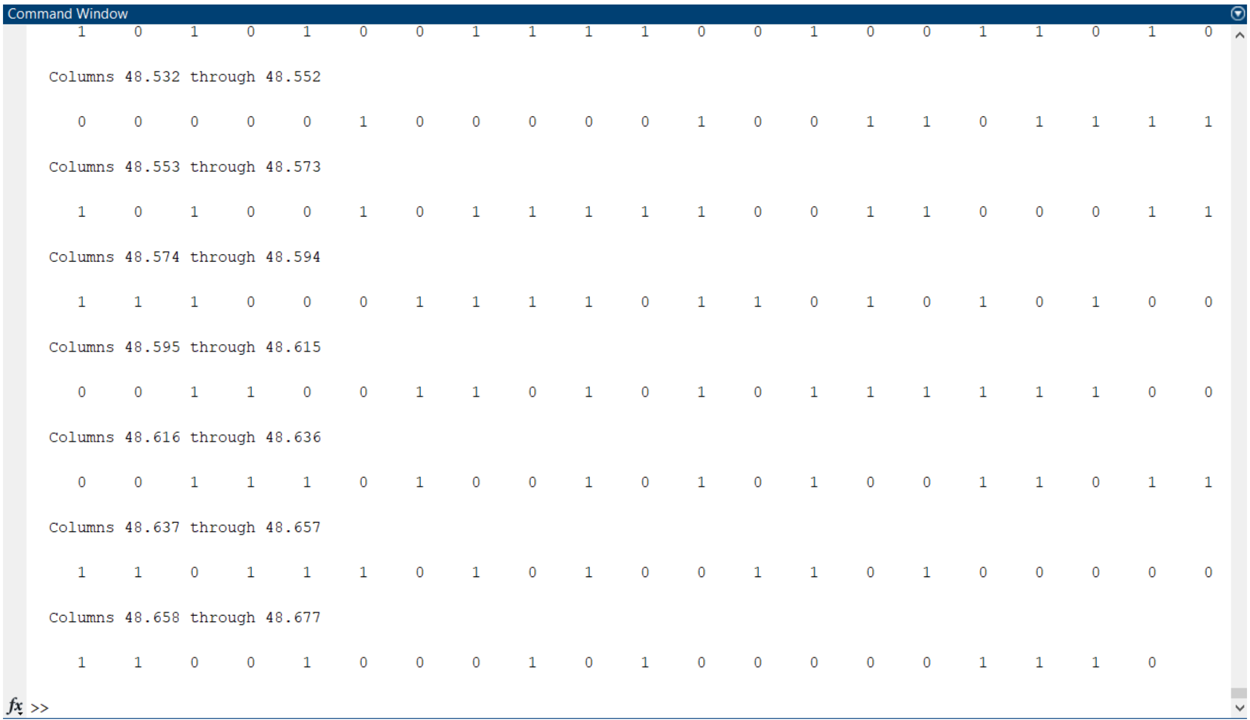
Ορίζω ως freq το διάνυσμα που περιέχει τις πιθανότητες των αγγλικών πεζών χαρακτήρων όπως μας δίνονται από τη σελίδα https://en.wikipedia.org/wiki/Letter_frequency, και nchar μεταβλητή που περιέχει το πλήθος των χαρακτήρων που παράγει η πηγή Α (10.000). Χρησιμοποιώ τη συνάρτηση randsample για τη δημιουργία της Πηγής Α, με ορίσματα τα freq, nchar. Χωρίζω το string που παράχθηκε, δημιουργώντας ένα cell array που περιέχει έναν χαρακτήρα ανά cell. Διαγράφω το τελευταίο cell που είναι κενό και προσθέτω ένα κενό δίπλα σε κάθε χαρακτήρα. Ύστερα το ξαναμετατρέπουμε σε ένα μεγάλο αλφαριθμητικό row, το οποίο έχει κενά ανάμεσα σε κάθε πεζό χαρακτήρα.



```
Command Window
callA =

  26×2 cell array

    {'a'}    {1×6 double}
    {'b'}    {1×9 double}
    {'c'}    {1×9 double}
    {'d'}    {1×5 double}
    {'e'}    {1×4 double}
    {'f'}    {1×4 double}
    {'g'}    {1×6 double}
    {'h'}    {1×4 double}
    {'i'}    {1×5 double}
    {'j'}    {1×6 double}
    {'k'}    {1×5 double}
    {'l'}    {1×4 double}
    {'m'}    {1×4 double}
    {'n'}    {1×6 double}
    {'o'}    {1×3 double}
    {'p'}    {1×5 double}
    {'q'}    {1×9 double}
    {'r'}    {1×5 double}
    {'s'}    {1×4 double}
    {'t'}    {1×5 double}
    {'u'}    {1×7 double}
    {'v'}    {1×6 double}
    {'w'}    {1×3 double}
    {'x'}    {1×5 double}
    {'y'}    {1×4 double}
    {'z'}    {1×9 double}
```

Σχήμα 4: Το λεξικό callA

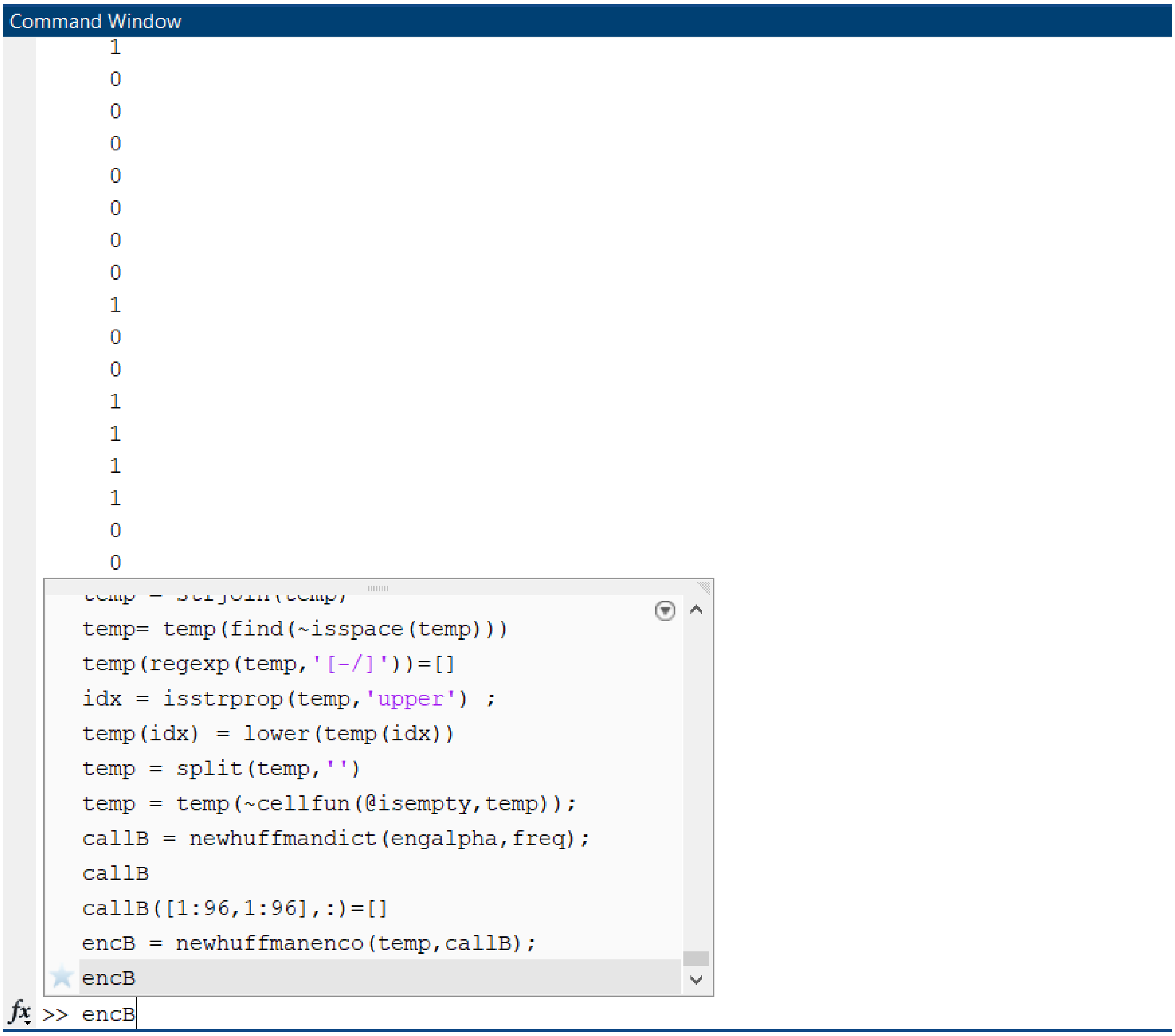
Δημιουργώ ένα cell array engalpha που περιέχει την αγγλική αλφάβητο σε πεζά γράμματα, προσέχοντας να έχω αντιστοίχιση χαρακτήρα και πιθανότητας στο freq. Αν δηλαδή έχω ως τρίτο γράμμα το r τότε στην τρίτη θέση του freq θα έχω την πιθανότητα εμφάνισης του r. Φτιαχνω ένα λεξικό callA με χρήση της συνάρτησης newhuffmandict, και ορίσματα τα engalpha και freq. Τέλος, κωδικοποιώ την Πηγή A χρησιμοποιώντας τη συνάρτηση newhuffmanenco με ορίσματα τα row και callA.



Σχήμα 5: Μέρος της κωδικοποιημένης πηγής encA

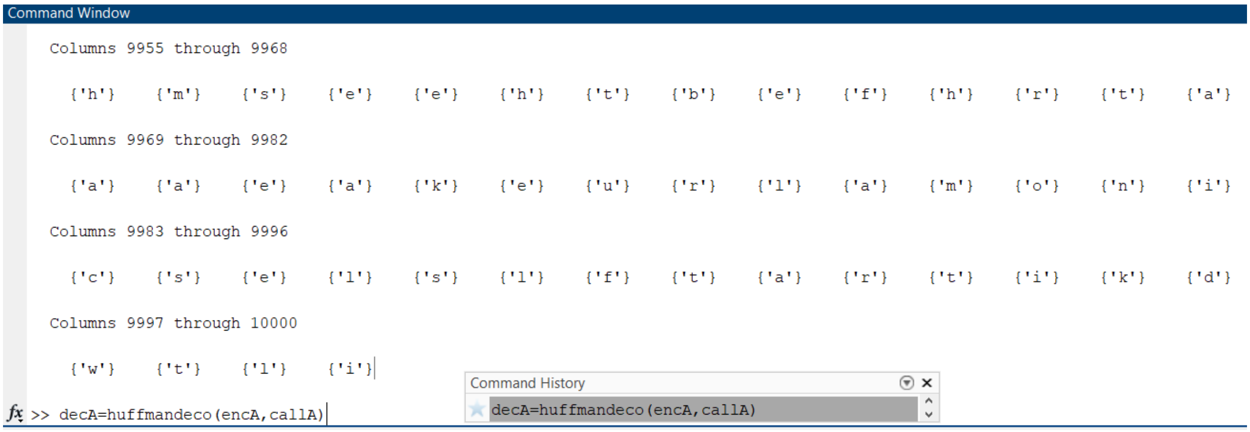
1.2.2 Κωδικοποίηση Πηγής Β

Ανοίγω το αρχείο 'kwords.txt', και το διαβάζω ως cell array με τη συνάρτηση textscan. Έπειτα, ενώνω όλες τις λέξεις που αναγνώστηκαν σε ένα μεγάλο αλφαριθμητικό temp. Με χρήση regular expressions, αφαιρώ χαρακτήρες όπως -,/ για τα οποία δε γνωρίζω πιθανότητες, καθώς και μετατρέπω τα διάφορα κεφαλαία γράμματα στα αντίστοιχα πεζά. Δημιουργώ ένα cell array το οποίο σε κάθε κελί έχει και από έναν χαρακτήρα. Φτιάχνω λεξικό callB με χρήση της συνάρτησης newhuffmandict, και ορίσματα τα engalpha και freq. Τέλος, κωδικοποιώ την Πηγή Β χρησιμοποιώντας τη συνάρτηση newhuffmanenco με ορίσματα τα temp και callB.



Σχήμα 6: Μέρος της κωδικοποιημένης πηγής encB

Με τη χρήση της huffmandeco, επιβεβαιώνω την ορθή αποκωδικοποίηση. Όπως φαίνεται στις εικόνες, η decA μου επιστρέφει ένα cell array με στοιχεία ίδια με αυτά του row. Η decB δε, μου επιστρέφει cell array ίδιο με το temp.



Σχήμα 7: Η αποκωδικοποίηση της encA έγινε σωστά

Σχήμα 8: Η αποκωδικοποίηση της encB έγινε σωστά

Το μέσο μήκος κώδικα και στις δύο κωδικοποιήσεις θα είναι το ίδιο, μιας και έχουν το ίδιο αλφάβητο και το ίδιο διάνυσμα πιθανοτήτων. Αυτό φαίνεται τρέχοντας το παρακάτω script, όπου προκύπτει ότι avgA ισούται με avbB. Βέβαια, η κωδικοποίηση της A έχει (48.671), θα 'χει μικρότερο μήκος από αυτή της B (144.415), μιας και κωδικοποιούμε μικρότερου μεγέθους πηγή.
Μέσο μήκος κώδικα L(W) N συμβόλων X και κωδικών λέξεων W_i : $\sum_{i=1}^{N} p(x_i)|W_i|$

Υπολογισμός μέσου μήκους κώδικα:

```
avgB=0;
avgA=0;
for i=1:1:26
    avgB = avgB + size(callB{i,2},2).*freq(i);
end

for i=1:1:26
    avgA = avgA + size(callA{i,2},2).*freq(i);
end
```

1.3 Ερώτημα 3

Αλλάζω το διάνυσμα πιθανοτήτων, βάζοντας αυτή τη φορά τις πιθανότητες των γραμμάτων με βάση την εμφάνιση τους στο αρχείο kwords. Φτιάχνω λεξικό callB3 με χρήση της συνάρτησης newhuffmandict, και ορίσματα τα engalpha και fkw. Τέλος, κωδικοποιώ την Πηγή B χρησιμοποιώντας τη συνάρτηση newhuffmanenco με ορίσματα τα temp και callB3.

Υπολογισμός μέσου μήκους κώδικα:

```
avgB3=0;
for i=1:1:26
    avgB3 = avgB3 + size(callB3{i,2},2).*fkw(i);
end
```

Παρατηρούμε πως το μέσο μήκος κώδικα για την πηγή B μεγαλώνει.
Υπολογίζω avgB3=5.2768

```
Command Window
callB3 =

  26×2 cell array

    {'a'}    {1×4  double}
    {'b'}    {1×6  double}
    {'c'}    {1×6  double}
    {'d'}    {1×6  double}
    {'e'}    {1×3  double}
    {'f'}    {1×7  double}
    {'g'}    {1×6  double}
    {'h'}    {1×5  double}
    {'i'}    {1×3  double}
    {'j'}    {1×9  double}
    {'k'}    {1×3  double}
    {'l'}    {1×4  double}
    {'m'}    {1×6  double}
    {'n'}    {1×4  double}
    {'o'}    {1×4  double}
    {'p'}    {1×6  double}
    {'q'}    {1×10 double}
    {'r'}    {1×4  double}
    {'s'}    {1×4  double}
    {'t'}    {1×4  double}
    {'u'}    {1×6  double}
    {'v'}    {1×7  double}
    {'w'}    {1×7  double}
    {'x'}    {1×10 double}
    {'y'}    {1×6  double}
fx  {'z'}    {1×8  double}
```

Σχήμα 9: Λεξικό callB3

1.4 Ερώτημα 4

Φτιάχνω νέο διάνυσμα πιθανοτήτων freqposs, παίρνοντας όλα τα πιθανά γινόμενα ανά δύο στοιχείων του διανύσματος freq. Το νέο μου αλφάβητο allcombs είναι ενα cell array, κάθε κελί του οποίου περιέχει όλους τους δυνατούς συνδυασμούς μηκους δύο των πεζών χαρακτήρων του αγγλικού αλφαβήτου. Προσομοιώνω τη δεύτερης επέκτασης πηγή Α με τη χρήση της συνάρτησης randsample με ορίσματα το allcombs, freqposs και το πλήθος των ζευγών προς κωδικοποίηση, δηλαδή 5.000.

```
{'md'}
{'he'}
{'yd'}
{'oo'}
{'sg'}
{'wf'}
{'ol'}
{'lf'}
{'oo'}
{'pl'}
{'cy'}
{'si'}
{'dx'}
{'hi'}
{'sf'}
{'ls'}
{'ae'}
{'ha'}
{'le'}
{'se'}
```

```
Command History
strext = randsample(allcombs,5000,true,freqposs);
```

Σχήμα 10: Η δεύτερης τάξης επέκταση πηγής A

```
Command Window
callAext =

  676×2 cell array

    {'aa'}    {1×12 double}
    {'ab'}    {1×16 double}
    {'ac'}    {1×15 double}
    {'ad'}    {1×11 double}
    {'ae'}    {1×10 double}
    {'af'}    {1×10 double}
    {'ag'}    {1×12 double}
    {'ah'}    {1×10 double}
    {'ai'}    {1×11 double}
    {'aj'}    {1×12 double}
    {'ak'}    {1×11 double}
    {'al'}    {1×10 double}
    {'am'}    {1×10 double}
```

Σχήμα 11: Το λεξικό για την κωδικοποίηση της δεύτερης τάξης επέκταση πηγής A

Υπολογισμός μέσου μήκους κώδικα:

```
avgAext=0;
for i=1:1:676
    avgAext = avgAext + size(callAext{i,2},2).*freqposs(i);
end
```

Βρίσκω avgAext=8.382. Παρατηρώ πως σε σχέση με το ερώτημα 2, το μέσο μήκος κώδικα έχει περίπου διπλασιαστεί. Αυτό βγάζει νόημα καθώς από τη θεωρία γνωρίζω ότι ισχύει $\overline{L} = \frac{\overline{L_n}}{n}$ και αρα $\overline{L} = \frac{\overline{L_2}}{2}$

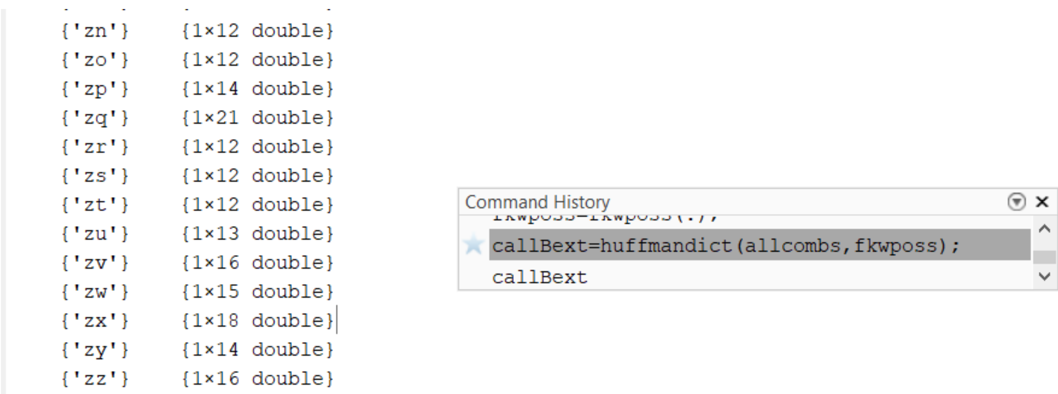
1.5 Ερώτημα 5

1.5.1 Κωδικοποίηση Πηγής Β με τις πιθανότητες ζευγών χαρακτήρων του ερωτήματος 4

Επαναλαμβάνω την ίδια διαδικασία που έκανα για το temp, για νέο string με τις λέξεις του kwords, temp2. Χωρίζω το string ανά ζεύγη χαρακτήρων, φτιάχνοντας έτσι ένα cell array A, κάθε κελί του οποίου περιέχει ένα ζεύγος χαρακτήρων. Κωδικοποιώ το A βάζοντας ως δεύτερο όρισμα στην huffmandict, το λεξικό του ερωτήματος 4 callAext.

1.5.2 Κωδικοποίηση Πηγής Β με τις πιθανότητες ζευγών χαρακτήρων του αρχείου kwords

Φτιάχνω νέο διάνυσμα πιθανοτήτων fkwposs, παίρνοντας όλα τα πιθανά γινόμενα ανά δύο στοιχείων του διανύσματος fkw. Κωδικοποιώ το A βάζοντας ως δεύτερο όρισμα στην huffmandict το λεξικό του ερωτήματος 4, callBext.



Σχήμα 12: Το λεξικό για την κωδικοποίηση της πηγής Β με τις πιθανότητες ζευγών χαρακτήρων του αρχείου kwords

Και στις δύο περιπτώσεις το μέσο μήκος κώδικα μεγαλώνει, ειδικά με τις πιθανότητες για ζεύγη χαρακτήρων από εκτίμηση του κειμένου.

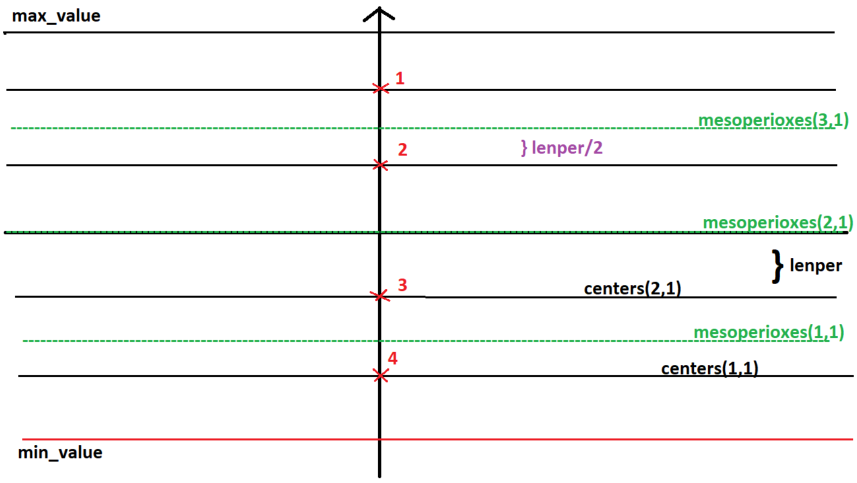
2 Κωδικοποίηση PCM

2.1 Ομοιόμορφος Κβαντιστής

Παρακάτω δίνεται η υλοποίηση ομοιόμορφου κβαντιστή στη MATLAB.

Ορίζω ως levels τα επίπεδα κβάντισης που ισούνται με 2^N. Όποιες τιμές είναι εκτός του εύρους min_value - max_value, θα τις θέτω ίσες με τις α-κραίες τιμές. Βρίσκω τα δείγματα που πέφτουν στα διάφορα τμήματα και τα αντιστοιχώ στα μέσα αυτών, τα centers. Ως centers αρχικοποιώ ένα διάνυσμα μήκους όσο το πλήθος των επιπέδων. Οι περιοχές είναι ισομήκεις με μήκος $lenper = (max\_value - min\_value)/levels$.

Οι μεσοπεριοχές είναι το μέσο της απόστασης ανάμεσα στα κέντρα δύο περιοχών. Ισούνται, δηλαδή, με την ελάχιστη επιτρεπτή τιμή συν το γινόμενο του μήκους των περιοχών που έχω διαπεράσει ήδη (πρώτη λούπα for). Τις αρχικοποιώ ως διάνυσμα μηδενικών μήκους όσο το πλήθος των επιπέδων μείον ένα. Στη μεταβλητή plmeso κρατώ το πλήθος αυτών των μεσοπεριοχών.



Σχήμα 13: Σχηματική αναπαράσταση του υλοποιημένου ομοιόμορφου κβαντιστή

Υπολογίζω τα κέντρα κάθε περιοχής μέσω ενός loop for, θέτοντας ως centers τις τιμές που προϋπολόγισα ως μεσοπεριοχές, μείον το μισό του μήκους μιας περιοχής. Το τελευταίο κέντρο το υπολογίζω έξω από τη λούπα, ως την τελευταία μεσοπεριοχή συν το μισό του μήκους της περιοχής.

Με μία λούπα for επαναλήψεων ίσων με το πλήθος των μεσοπεριοχών, βρίσκω την μεσοπεριοχή στην οποία έπεσαν τα δείγματα x. Βρίσκοντας τη θέση τους thesi, ορίζω ως κβαντισμένο σήμα εξόδου xq τα στοιχεία του διανύσματος centers με δείκτη τη μεταβλητή thesi.

Ο κώδικας του ομοιόμορφου κβαντιστή

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

levels=2^N;
centers=zeros(levels,1);
lenper=2*max_value/levels;

mesoperioxes=zeros(levels-1,1);

plmeso=length(mesoperioxes);

for i=1:1:plmeso
    mesoperioxes(i,1)= min_value + i*lenper;
```

```
end

for j=1:1:plmeso
    centers(j,1)=mesoperioxes(j,1)-lenper/2;
end

centers(levels,1)=mesoperioxes(plmeso,1)+lenper/2;

deigmata=length(x);

for k=1:1:deigmata
    thesi=binarysearch(x(k,1),mesoperioxes,plmeso);
    xq(k,1)=centers(thesi,1);
end

function deik= binarysearch(val,mesoperioxes,plmeso)
low=1;
high= plmeso + 1;
deik=fix((low+high)/2);
while(low<high)&&(deik~=1)
    if (val >= mesoperioxes(deik-1,1)) && (val<mesoperioxes(deik,1))
        return;
    elseif val >= mesoperioxes(deik,1)
        low=deik;
    else
        high=deik-1;
    end
    deik=fix((low+high)/2);
    if(high-low)==1
        deik=high;
        return;
    end
end
deik=low;
return;
return;
```

2.2 Μη ομοιόμορφος κβαντιστής Lloyd-Max

Ο μη ομοιόμορφος κβαντιστής έχει ομοιότητες με τον ομοιόμορφο στη βάση της υλοποίησής του. Τα επίπεδα και οι μεσοπεριοχές ορίζονται με τον ίδιο τρόπο.

Έχω όμως πλέον και την αναμενόμενη τιμή, την οποία αρχικοποιώ ως διάνυσμα μήκους όσα και τα επίπεδα. Με μια for loop ορίζω τις αναμενόμενες τιμές όπως τα κέντρα του ομοιόμορφου κβαντιστή.

Βρίσκω όπως και προηγουμένως, τη θέση στην οποία βρέθηκε το δείγμα. Κρατάω σε ένα διάνυσμα emfaniseis πόσες φορές έχει αντιστοιχηθεί δείγμα σε συγκεκριμένο επίπεδο κβάντισης. Αυτό θα μου χρειαστεί για επόμενο ερώτημα. Ύστερα, υπολογίζω τα κέντρα μάζας για κάθε θέση και ορίζω ως διάνυσμα εξόδου για δείκτη j από ένα έως τον αριθμό δειγμάτων, την τιμή στο διάνυσμα της αναμενόμενης τιμής με δείκτη τη θέση που βρήκαμε στην ίδια επανάληψη.

Έπειτα, υπολογίζω την αναμενόμενη τιμή για την επόμενη επανάληψη. Εάν το κέντρο μάζας δεν είναι μηδέν (στην οποία περίπτωση ορίζω αναμενόμενη τιμή τη μεσοπεριοχή της τωρινής επανάληψης), η αναμενόμενη τιμή θα ισούται με το κέντρο μάζας της επανάληψης j διά το πλήθος των εμφανίσεων για το επίπεδο κβάντισης που ελέγχω.

Ξαναυπολογίζω τις μεσοπεριοχές, οι οποίες αυτη τη φορά θα ισούνται με την αναμενόμενη τιμή συν την επόμενη αναμενόμενη τιμή δυα δύο, σύμφωνα με τους τύπους που μας δίνονται. Τέλος, ορίζω την παραμόρφωση ως $D_i = mean((x - xq)^2)$. Η επανάληψη όλης αυτής της διαδικασίας θα τελειώσει, όταν $|D_i - D_{i-1}| < e$.

Ο κώδικας του μη ομοιόμορφου κβαντιστή

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

D(1,1)=1;
levels=2^N;
lenper=(max_value-min_value)/levels;
mesoperioxes=zeros(levels-1,1);
plmeso=length(mesoperioxes);
epsilon=10^(-16);

for j=1:1:plmeso
    mesoperioxes(j,1)= min_value + j*lenper;
end

anamenomeni= zeros(levels,1);

for j=1:1:plmeso
    anamenomeni(j,1)=mesoperioxes(j,1)- lenper/2;
end

```

```
anamenomeni(levels,1)=mesoperioxes(plmeso,1)+lenper/2;
deigmata = length(x);

for i= 2:1:10^5
    emfaniseis = zeros(levels,1);
    kentraMazas = zeros(levels,1);

    for j=1:1:deigmata
        thesi = binarysearch(x(j,1),mesoperioxes,plmeso);
        emfaniseis(thesi,1)=emfaniseis(thesi,1) + 1;
        kentraMazas(thesi,1)=kentraMazas(thesi,1)+ x(j,1);
        xq(j,1)=anamenomeni(thesi,1);
    end

    for j=1:1:levels
        if kentraMazas(j,1)==0
            if j==1
                anamenomeni(j,1)=mesoperioxes(j,1);
            else
                anamenomeni(j,1)=mesoperioxes(j-1,1);
            end
        else
            anamenomeni(j,1)=kentraMazas(j,1)/emfaniseis(j,1);
        end
    end

    for j=1:1:plmeso
        mesoperioxes(j,1)=(anamenomeni(j,1) + anamenomeni(j+1,1))
            /2;
    end

    D(i,1)=mean((x-xq).^2);
    if abs(D(i,1)-D(i-1,1))<epsilon
        centers=anamenomeni;
        return;
    end
end

function deik= binarysearch(val,mesoperioxes,plmeso)
low=1;
high= plmeso + 1;
deik=fix((low+high)/2);
while(low<high)&&(deik~=1)
```

```
    if (val >= mesoperioxes(deik-1,1)) && (val<mesoperioxes(deik,1)
        )
        return;
    elseif val >= mesoperioxes(deik,1)
        low=deik;
    else
        high=deik-1;
    end
    deik=fix((low+high)/2);
    if(high-low)==1
        deik=high;
        return;
    end
end
deik=low;
return;
return;
```

2.3 Ερώτημα 1

2.3.1 α. Υπολογισμός SQNR

Κωδικοποιώ την πηγή Α για min_value=0, max_value=4, N=4,6 bits. Για τον θεωρητικό υπολογισμό του SQNR παίρνω τον τύπο: $SQNR_{db} = 1,76 + 6,02NdB$. Καταλήγω στις τιμές:

Για N=4: 25,84 dbs
Για N=6: 37,88 dbs

Στην πράξη, μπορούμε να επιβεβαιώσουμε πως το σφάλμα κβάντισης εμφανίζεται σε ομοιόμορφη κατανομή, όταν το μέγεθος του βήματος είναι κατά πολύ μικρότερο από το δυναμικό εύρος των δειγμάτων του σήματος και τα δείγματα είναι αρκετά. Επομένως, για τον πειραματικό υπολογισμό του SQNR χρησιμοποιώ τον τύπο: $SQNR = mean(x^2/D)$. Έτσι καταλήγω στις λίγο διαφορετικές τιμές:

Για N=4: 88.5865 ή 19.4736754331 dbs
Για N=6: 1.1436ε+03 ή 30.5827414669 dbs

Βλέπουμε ότι οι μετρήσεις μας είναι κοντά με τις αντίστοιχες θεωρητικές προσεγγίσεις.

2.3.2 b. distortion overload

Αφού παίρνουμε τις απόλυτες τιμές του t, αποκλείεται να φύγουμε κάτω από την ελάχιστη τιμή που είναι 0. Κανόντας δοκιμές, υπολογίζω πόσα δείγματα από τα 10.000 ξεπερνάνε την τιμή 4. Βρίσκω:

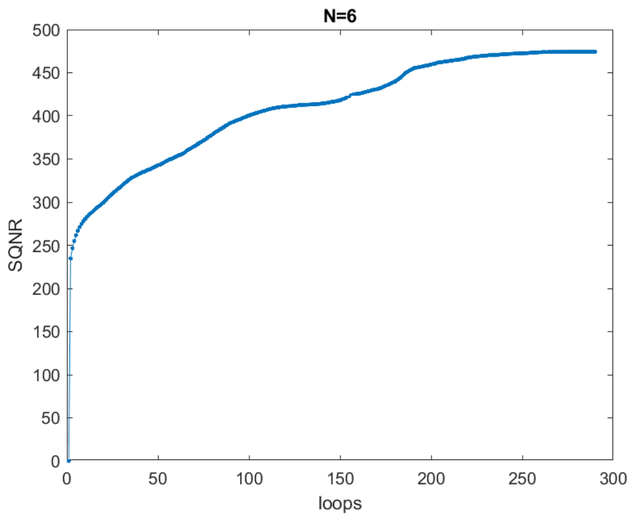
Για N=4: κυμαίνεται η πιθανότητα περίπου από 1,67% έως 1,94%
Για N=6: κυμαίνεται η πιθανότητα περίπου από 0,17% έως 0,31%

```
%PHGH A
t = (randn(10000,1)+j*randn(10000,1))/sqrt(2);
x= abs(t) .^ 2;

[xq,centers]=my_quantizer(x,4,0,4)
[xq,centers]=my_quantizer(x,6,0,4)

%PHGH B

[y, fs] = audioread('speech.wav');
info = audioinfo('speech.wav');
N = info.BitsPerSample;
sound(y, fs);


%calculate SQNR:
D=mean((x-xq).^2);
SQNR=mean(x.^2)/D;
```

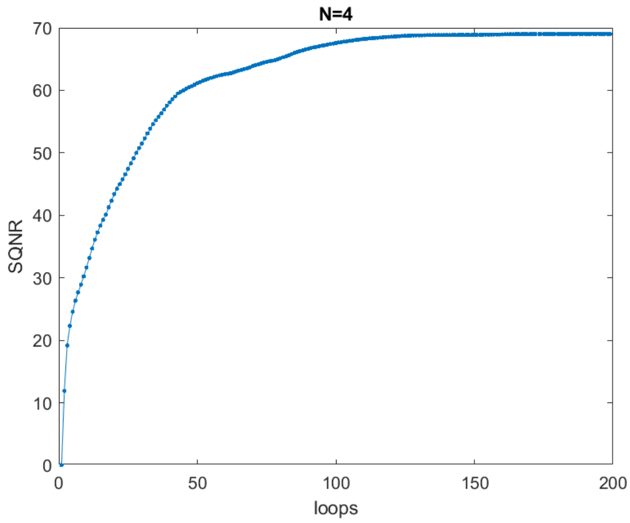
2.4 Ερώτημα 2

2.4.1 α.

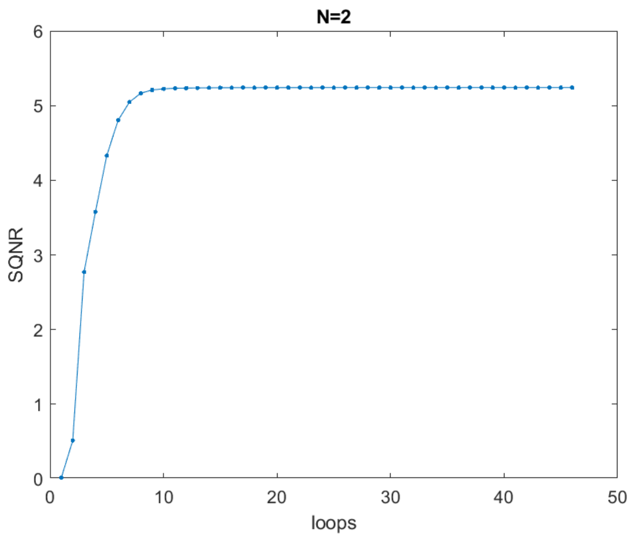
Κωδικοποιώ την πηγή B με μη-ομοιόμορφο κβαντιστή για min_value=-1, max_value=1, N=2,4,6 bits. Όσο αυξάνω το N, τόσες επαναλήψεις Kmax χρειαζόμαστε για τον τερματισμό του αλγορίθμου. Για λίγα bits περιμένω να έχω μεγάλη παραμόρφωση η οποία αλλάζει πολύ λίγο ύστερα, επομένως θα φτάσω στην τιμή που καθορίζει η μεταβολή της παραμόρφωσης γρήγορα. Με το παρακάτω script παίρνω γραφικά την αλλαγή του SQNR για διαφορετικά N.



Σχήμα 14: SQNR for N=6



Σχήμα 15: SQNR for N=4



Σχήμα 16: SQNR for N=2

```
SQNRLloydMax=zeros(3,1);
plot_title=['N=2';'N=4';'N=6'];
xq=zeros(length(y),3);
axis_x=[1:1:size(y)];

for i=2:2:6
    [xq(:,i/2),Centers,D]=Lloyd_Max(y,i,-1,1);
    kmax=length(D);
    loops=[1:1:kmax];
    sqnr_total=zeros(kmax,1);

for j=1:1:kmax
    sqnr_total(j,1)=mean(y.^2)/D(j,1);
end

figure;
plot(loops,sqnr_total,'.-')
title(plot_title(i/2,:));
xlabel('loops');
ylabel('SQNR');
SQNRLloydMax(i/2,1)=sqnr_total(kmax,1);

end
```

2.4.2 β.

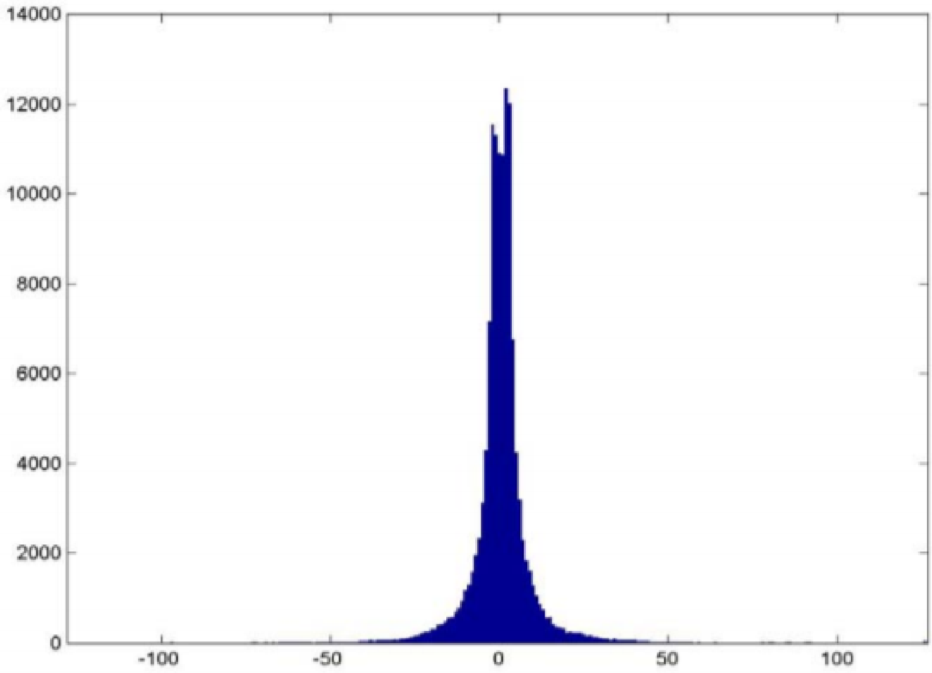
Κωδικοποιώ την πηγή B με ομοιόμορφο κβαντιστή για min_value=-1, max_value=1, N=2,4,6 bits. Υπολογίζω το SQNR με τον ίδιο τρόπο όπως και στο ερώτημα 1α. Βρίσκω:

Για N=2: σχεδόν 0 dbs
Για N=4: 10.75 dbs
Για N=6: 23.7 dbs

Γνωρίζοντας την εξής σχέση: $H(X) \leq \overline{R} \leq H(X) + \frac{1}{n}$, και από τα αποτελέσματά μου, παρατηρώ πως όσο μεγαλώνουν τα N, έχω καλύτερα αποτέλεσματα. Ο μη-ομοιόμορφος κβαντιστής Lloyd-Max είναι πιο αποδοτικός, καθώς παρά τους περισσότερους υπολογισμούς και την πιο πολύπλοκη υλοποίηση, κβαντίζει το σήμα με τη λιγότερη δυνατή παραμόρφωση. Από την άλλη, ο ομοιόμορφος κβαντιστής έχει το μεγαλύτερο σφάλμα μετά την κβάντιση, καθώς οι στάθμες κβάντισης είναι ισομήκεις και ανεξάρτητες του σήματος εισόδου.

2.4.3 γ.

Όσον αφορά την πιθανότητα εμφάνισης κάθε στάθμης του κβαντιστή, είναι σημαντικό να θυμόμαστε πως τα δείγματα του σήματος φωνής δεν είναι κατανεμημένα ομοιόμορφα σε όλο το δυνατό εύρος τιμών τους. Όπως φαίνεται στο παρακάτω ιστόγραμμα για αρχείο φωνής των 16 bits ανά δείγμα μετά από ομοιόμορφη κβάντιση 8 bit ανά δείγμα, σχεδόν όλο το σήμα είναι τοποθετημένο σε 80 από τα 256 επίπεδα. Ο μη ομοιόμορφος κβαντιστής όχι μόνο χρησιμοποιεί περισσότερα επίπεδα κβάντισης, αλλά λαμβάνει και υπόψη την pdf του σήματος.



Σχήμα 17: Ιστόγραμμα σήματος φωνής

Επεξεργαζόμαστε τη συνάρτηση Lloyd-Max, προκειμένου να επιστρέφεται ως έξοδος και το διάνυσμα emfaniseis. Αυτό μας δείχνει πόσα δείγματα αντιστοιχήθηκαν σε κάθε επίπεδο κβάντισης. Για να υπολογίσω την εντροπεία, τρέχω το παρακάτω σκριπτάκι:

```
emf_sum = sum(emfaniseis)
p(:)=emfaniseis(:)/emf_sum;
entropy=0;
%To M tha pernei kathe fora timh 2^N (gia N=2 -> M=4)
for i=1:1:M
     entropy = entropy-p(i)*log2(p(i));
end
```

Βρίσκω:

Για N=2: emfaniseis = 2373 7755 25234 4559, entropy = 1.4771
Για N=4: emfaniseis = 142 297 905 1683 2258 2715 4018 11034 7026 3299 2532 1894 1248 553 259 58, entropy = 3.2443
Για N=6: emfaniseis = 0 0 0 ... 7 5 6 2 1, entropy = NaN

2.4.4 δ.

Το πιο διαδεδομένο μέτρο εκτίμησης της ποιότητας ομιλίας είναι ο λόγος του σήματος προς θόρυβο, το οποίο με τη σειρά του χρησιμοποιεί ως μέτρο διαταραχής το μέσο τετραγωνικό σφάλμα που δίνεται από τον τύπο: $d[\overline{x},\overline{y}] = \frac{1}{N} \cdot \sum_{k=1}^{N}[x_k - y_k]^2$

```
for i=1:1:N
    a(i)=(y(i)-xq(i))^2;
end
sum_a=sum(a);
mse=sum_a/N;
```

Ομοιόμορφος κβαντισμός:
Το βήμα κβαντισμού είναι αρκετά μικρό, έτσι το σφάλμα κβαντισμού κατανέμεται ομοιόμορφα σε κάθε περιοχή κβαντισμού. Αυτό συνήθως είναι καλή προσέγγιση για πλήθος επιπέδων κβάντισης μεγαλύτερο ίσο του 64.
Τρέχοντας το script παίρνω:

Για N=2: mse=0.0592
Για N=4: mse=0.0031
Για N=6: mse=1.2019e-04

Μη-ομοιόμορφος κβαντισμός: Τρέχοντας το script παίρνω:

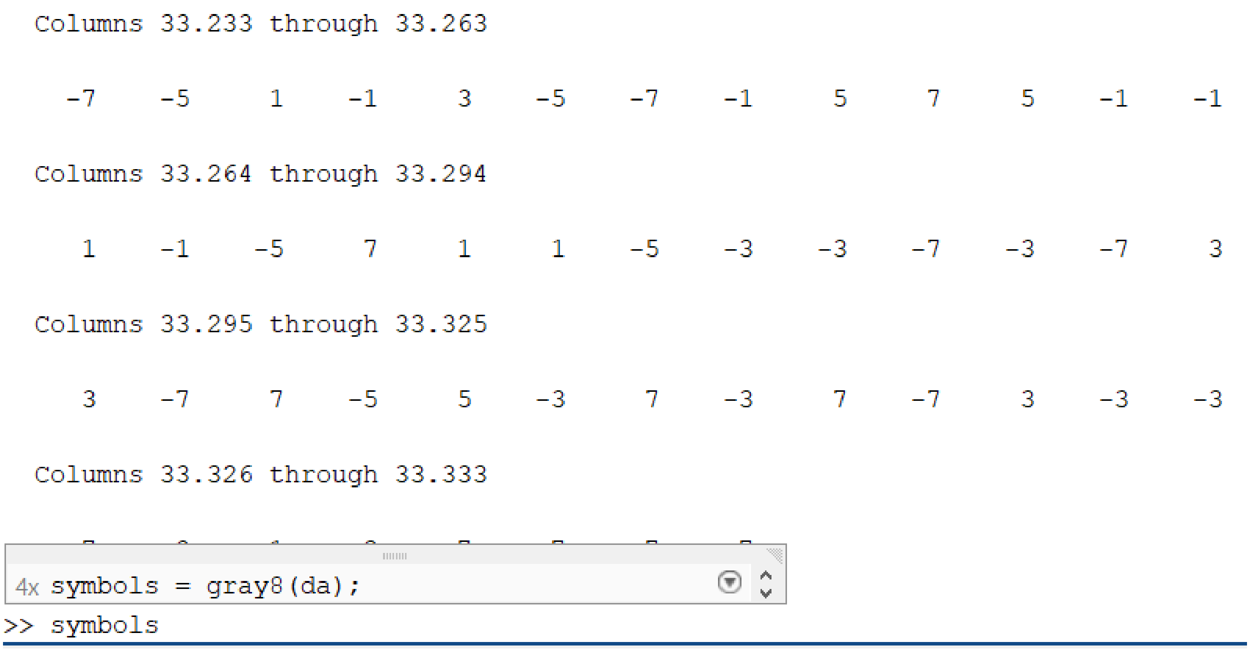
Για N=2:mse=6.9978e-04
Για N=2:mse=4.7473e-05
Για N=2:mse=1.2018e-05

Βλέπω ότι οι τιμές που παίρνω εδώ ως προς το σφάλμα είναι σημαντικά μικρότερες!

2.5 Ερώτημα 3

2.5.1 MAPPER

Φτιάχνω συναρτήσεις binary2, gray2, binary4, gray4, binary8, gray8, binary16, gray16. Η κάθε μία κάνει την ανάλογη αντιστοίχιση δυαδικών ψηφίων σε δεκαδικούς αριθμούς, άναλογα με το μέγεθος της κβάντισης (αριθμός στην ονομασία τους) και αν υλοποιούν gray κωδικοποίηση ή όχι.



Σχήμα 18: Δοκιμή συνάρτησης gray8 για gray=1, M=8

2.5.2 Διαμορφωτής M-PAM

Ο διαμορφωτής που έφτιαξα μέσω MATLAB παίρνει ως είσοδο το διάνυσμα symbols που προκύπτει στο προηγούμενο ερώτημα και το μέγεθος της διαμόρφωσης και βγάζει ως έξοδο το διαμορφωμένο σήμα. Ορίζω τις μεταβλητές Tsymbol, fsymbol, Tsample, Tc, fc, Es σύμφωνα με τις οδηγίες που μας δίνονται στην εκφώνηση, στις χρονικές μονάδες προσομοίωσης. Με δύο for loops κανω τον πολλαπλασιασμό:

$$s_m(t) = s_m * g_T(t) * cos(2\pi f_c t),\ 0 \leq t \leq T_{symbol}$$
$$s_m = (2m - 1 - M) * A,\ m = 1, ..., M$$

2.5.3 Κανάλι AGWN

Για την προσομοίωση καναλιού προσθετικού λευκού θορύβου, δίνω στην συνάρτηση ορίσματα το διαμορφωμένο σήμα του προηγούμενου ερωτήματος, την τιμή SNR, και το M. Για ενέργεια ανά bit $E_b = \frac{E_s}{log_2 M}$ βρίσκω από τον τύπο $SNR = 10 * log_{10}(\frac{E_b}{N_0})$ το N_0(5). Η γκαουσιανή κατανομή που φτιάχνουμε θα έχει μέση τιμή μηδέν και διασπορά $s = \sqrt{\frac{N_0}{2}}$(8). Για να προσθέσω θόρυβο σε κάθε δείγμα του διαμορφωμένου σήματος, πρέπει η συνάρτηση randn να έχει ως ορίσματα τις διαστάσεις του sm.

```
function [r,N0] = awgn(sm,SNR,M)

Es = 1;
Eb = Es / log2(M);
N0 = Eb/(10^(SNR/10));

[m, n] = size(sm);
noise = sqrt(N0 / 2) * randn(m,n);

r = sm + noise;
```

2.5.4 Αποδιαμορφωτής M-PAM

Σύμφωνα με τη θεωρία, η ενέργεια του ορθογώνιου παλμού g είναι:
$E_g = \int_0^{T_{symbol}} A^2 dt = A^2 T_{symbol}$

. Επειδή το σύνολο των PAM σημάτων είναι διάστασης N=1, υπάρχει μόνο μια συνάρτηση βάσης y(t), η οποία δίδεται από την(6):

. $y(t) = \frac{1}{\sqrt{T_{symbol}}},\ 0 \leq t \leq T_{symbol}$

Άρα η έξοδος του αποδιαμορφωτή(9), είναι:

$r = \frac{1}{\sqrt{T_{symbol}}} \int_0^{T_{symbol}} r(t) dt$

```
function rnew = demodulator(r)

Tsymbol = 40;

for t = 1: Tsymbol
    y(t, 1) = 1/sqrt(Tsymbol);
end

rnew = (r * y);
```

2.5.5 Φωρατής M-PAM

Ο φωρατής θα παίρνει ως είσοδο το αποδιαμορφωμένο σήμα - διάνυσμα r και βρίσκει ποιο σύμβολο έχει σταλθεί. Η εκφώνηση μας λέει ότι για $M > 2$ θέλω η μέση ενέργεια συμβόλων να είναι 1, άρα υπολογίζω την ενέργεια του παλμού ανάλογα(3). Στη γραμμή (10) υπολογίζω κάθε πιθανό σύμβολο που θα μπορούσε να σταλθεί μέσω του τύπου:

$$s_m = \sqrt{E_g} A_m,\ m = 1, .., M$$

Ύστερα, στη γραμμή 16, υπολογίζω την απόσταση των δύο γειτονικών στοιχείων r με όλα τα πιθανά σύμβολα που μπορεί να στάλθηκαν και τα αποθηκεύω στο διάνυσμα tmp. Ως σύμβολο που διαλέγω τελικά ως το απεσταλμένο κρατώ αυτό που έχει τη μικρότερη απόσταση(18).

```
1  function symbols1 = foratis(rnew,M)
2
3  if M>2
4      Eg=1;
5  else
6      Eg=3/M^2-1;
7  end
8
9  for m = 1: M
10     s(m, 1) = sqrt(Eg/2)*(2*m-1-M);
11 end
12
13 [rsize, ~] = size(rnew);
14 for j =1: rsize-1
15     for m = 1: M
16         tmp(m, 1) = norm([rnew(j,1),rnew(j+1,1)] - s(m,:));
17     end
18 [~, symbols1(j, 1)] = min(tmp);
19 end
20
21 symbols1 = mod(symbols1,M);
```

2.5.6 DEMAPPER

Ο χρήστης επιλέγει τα κατάλληλα ορίσματα για το M (2,4,8,16) και το gray(=0 αν θέλω δυαδική αποκωδικοποίηση) ανάλογα με την κωδικοποίηση που έχει επιλέξει προηγουμένως. Για N=2, δεν έχει σημασία αν διάλεξα gray=1 ή όχι καθώς η κωδικοποίηση είναι η ίδια.

3 Πηγαίος κώδικας

3.1 Ερώτημα 1

3.1.1 Ζητούμενο 1

α.

```
function finalcode = newhuffmandict(s,p)

if (ndims(p) ~= 2) | (min(size(p))>1) | ~isreal(p) | ~isnumeric(p)
error('O p prepei na einai arithmitiko dianusma pragmatikwn
    arithmwn');
end
if ~isa(s, 'cell')
    error('To alphabhto symbolwn prepei na einai ths morfhs cell
        array')
end
if length(p) ~= length(s)
    error('Prepei to plithos twn stoixeiwn toy dianusmatos
        pithanothtwn kai tou alphabhtou symbolwn na symfwnoun')
end

global finalcode
finalcode = cell(length(p),2);
if length(p) > 1
%p = p/sum(p);
s = sorting(s,p);
traversetree(s, []);
else
finalcode = {'1'};
end

function s = sorting(s,p)

while numel(s) > 2
[p, idx] = sort(p);
p(2) = p(1) + p(2);
p(1) = [];
s = s(idx);
s{2} = {s{1},s{2}};
```

```
s(1) = [];
end


function traversetree(node, word)

global finalcode
if isa(node,'cell')
traversetree(node{1},[word 0]);
traversetree(node{2}, [word 1]);
else
finalcode{node,1} = node;
%finalcode{node,2} = char('0' + word);
finalcode{node,2} = word;
%each Huffman codeword is represented as a row vector
end
```

β.

```
function encoded = newhuffmanenco(shma, dict)

[m,n] = size(shma);

if (m==1 || n==1)
  [m,n] = size(shma);
  shma = mat2cell(shma, ones(1,m), ones(1,n) );
end



largeSz = 0;
dictSz = size(dict,1);

for i = 1 : dictSz
    tmp = size(dict{i,2},2);
    if (tmp > largeSz)
        largeSz = tmp;
    end
end

encoded = zeros(1, length(shma)*largeSz);

index = 1;
for i = 1 : length(shma)
```

```
    ithcode = [];
    for j = 1 : dictSz
        if( shma{i} == dict{j,1} )
            ithcode = dict{j,2};
            break;
        end
    end

    ithcodelgth = length(ithcode);
    encoded(index : index + ithcodelgth - 1) = ithcode;
    index = index + ithcodelgth;
end

encoded = encoded(1:index-1);

if( n == 1 )
    encoded = encoded';
end
```

γ.

```
function [decoded] = newhuffmandeco(encoded,dict)

size_of_dic = size(dict,1);

function [dictEncosz,symbol] = find(kwdikas)
symbol=[];


for j = 1:size_of_dic
    dictEnco = cell2mat(dict(j,2));
    dictEncosz = size(dictEnco, 2);
   size = size(kwdikas, 2);
    if dictEncosz > size
        break;
    end
    if isequal(size,dictEncosz) && isequal(dictEnco,kwdikas)
        symbol = cell2mat(dict(j,1));
        break;
    end
end
end

start = 1;
```

```
finish = 1;
decoded = [];
[n1, n2] = size(encoded);
if n1 > n2
    n = n1;
else
    n = n2;
end

while finish < n
   [dictEncosz,symbol] = find(encoded(start:finish));

    decoded =[decoded symbol];

    start = finish + 1;
    finish = finish + dictEncosz;

end
```

3.1.2 Ζητούμενο 2

α.

```
nchar = 10000;
freq = [ 8.167 1.492 2.782 4.253 12.702 2.228 2.015 6.094 6.966 ...

    0.153 0.772 4.025 2.406 6.749 7.507 1.929 0.095 5.987 6.327 ...
    9.056 2.758 0.978 2.36 0.15 1.974 0.075]./100;

str = char(randsample('a':'z',nchar,true,freq));

disp(str)
str = split(str,'')
str = str(~cellfun(@isempty,str));
str = strcat(str,{' '});
row = cell2mat(str.');

engalpha = {'e';'a';'r';'i';'o';'t';'n';'s';'l';'c';'u';'d';'p';'m'
    ;'h';'g';'b';'f';'y';'w';'k';'v';'x';'z';'j';'q'};
callA = newhuffmandict(engalpha,freq);
callA([1:96,1:96],:)=[]
encA = newhuffmanenco(row,callA);
```

β.

```
fid = fopen('kwords.txt');
data = textscan(fid,'%s');
fclose(fid);
temp = data{1}(:,1)
temp = strjoin(temp)
temp= temp(find(~isspace(temp)))
temp(regexp(temp,'[-/]'))=[]
idx = isstrprop(temp,'upper') ;
temp(idx) = lower(temp(idx))
temp = split(temp,'')
temp = temp(~cellfun(@isempty,temp));
callB = newhuffmandict(engalpha,freq);
callB([1:96,1:96],:)=[]
encB = newhuffmanenco(temp,callB);
```

```
avgB=0;
avgA=0;
for i=1:1:26
    avgB = avgB + size(callB{i,2},2).*freq(i);
end
```

```
for i=1:1:26
    avgA = avgA + size(callA{i,2},2).*freq(i);
end
```

3.1.3 Ζητούμενο 3

```
fkw=[15.62 10.53 9.05 8.72 7.15 6.29 5.84 5.17 5.14 5 2.78 2.29
    2.27 2.13 2.12 2.01 1.93 1.89 1.47 0.79 0.68 0.44 0.41 0.2 0.07
     0.01]/100;
engkwords = {'k';'e';'i';'a';'n';'o';'s';'t';'r';'l';'h';'c';'d';'
    u';'m';'g';'y';'p';'b';'w';'f';'v';'z';'j';'x';'q'};
callB3 = newhuffmandict(engkwords,fkw);
callB3([1:96,1:96],:)=[]
encB3 = newhuffmanenco(temp,callB3);
```

```
avgB3=0;
for i=1:1:26
    avgB3 = avgB3 + size(callB3{i,2},2).*fkw(i);
end
```

3.1.4 Ζητούμενο 4

```
freq1=[1.492 0.095 0.153 4.025 8.167 5.987 1.929 7.507 4.253 1.974
     2.758 6.966 6.749 2.015 12.702 2.406 0.075 2.782 6.094 2.228
    0.772 0.978 9.056 2.36 6.327 0.15]./100;
freqposs=[freq1(:).*(freq1(:))'];
freqposs=freqposs(:);
allcombs={
    'aa';
    'ab';
    'ac';
    'ad';
    'ae';
    ...
    'zz';}

strext = randsample(allcombs,5000,true,freqposs);
callAext=huffmandict(allcombs,freqposs);
encAext=huffmanenco(strext,callAext);
```

```
avgAext=0;
for i=1:1:676
    avgAext = avgAext + size(callAext{i,2},2).*freqposs(i);
end
```

3.1.5 Ζητούμενο 5

α.

```
temp2 = data{1}(:,1)
temp2 = strjoin(temp2)
temp2= temp2(find(~isspace(temp2)))
temp2(regexp(temp2,'[-/]'))=[]
idx2 = isstrprop(temp2,'upper')
temp2(idx2) = lower(temp2(idx2))
ns = numel(temp2);
n=2;
A = cellstr(reshape([temp2 repmat(' ',1,ceil(ns/n)*n-ns)],n,[])')'
A(14559)=[]
encBextA=huffmanenco(A,callAext);
```

β.

```
fkw1=[8.72 1.47 2.29 2.27 10.53 0.68 2.01 2.78 9.05 0.2 15.62 5
    2.12 7.15 6.29 1.89 0.01 5.14 5.84 5.17 2.13 0.44 0.79 0.07
    1.93 0.41]./100;
fkwposs=[fkw1(:).*(fkw1(:))'];
fkwposs=fkwposs(:);
callBext=huffmandict(allcombs,fkwposs);
encBextB=huffmanenco(A,callBext);
```

3.2 Ερώτημα 2

3.2.1 Ζητούμενο 1

```
%PHGH A
t = (randn(10000,1)+j*randn(10000,1))/sqrt(2);
x= abs(t) .^ 2;

[xq,centers]=my_quantizer(x,4,0,4)
[xq,centers]=my_quantizer(x,6,0,4)
```

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

levels=2^N;
centers=zeros(levels,1);
lenper=2*max_value/levels;

mesoperioxes=zeros(levels-1,1);

plmeso=length(mesoperioxes);

for i=1:1:plmeso
    mesoperioxes(i,1)= min_value + i*lenper;
end

for j=1:1:plmeso
    centers(j,1)=mesoperioxes(j,1)-lenper/2;
end

centers(levels,1)=mesoperioxes(plmeso,1)+lenper/2;

deigmata=length(x);

for k=1:1:deigmata
```

```
    thesi=binarysearch(x(k,1),mesoperioxes,plmeso);
    xq(k,1)=centers(thesi,1);
end

function deik= binarysearch(val,mesoperioxes,plmeso)
low=1;
high= plmeso + 1;
deik=fix((low+high)/2);
while(low<high)&&(deik~=1)
    if (val >= mesoperioxes(deik-1,1)) && (val<mesoperioxes(deik,1)
        )
        return;
    elseif val >= mesoperioxes(deik,1)
        low=deik;
    else
        high=deik-1;
    end
    deik=fix((low+high)/2);
    if(high-low)==1
        deik=high;
        return;
    end
end
deik=low;
return;
return;
```

α.

```
%calculate SQNR:
D=mean((x-xq).^2);
SQNR=mean(x.^2)/D;
```

3.2.2 Ζητούμενο 2

```
%PHGH B
[y, fs] = audioread('speech.wav');
info = audioinfo('speech.wav');
N = info.BitsPerSample;
sound(y, fs);
```

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

```

```
D(1,1)=1;
levels=2^N;
lenper=(max_value-min_value)/levels;
mesoperioxes=zeros(levels-1,1);
plmeso=length(mesoperioxes);
epsilon=10^(-16);

for j=1:1:plmeso
    mesoperioxes(j,1)= min_value + j*lenper;
end

anamenomeni= zeros(levels,1);

for j=1:1:plmeso
    anamenomeni(j,1)=mesoperioxes(j,1)- lenper/2;
end

anamenomeni(levels,1)=mesoperioxes(plmeso,1)+lenper/2;
deigmata = length(x);

for i= 2:1:10^5
    emfaniseis = zeros(levels,1);
    kentraMazas = zeros(levels,1);

    for j=1:1:deigmata
        thesi = binarysearch(x(j,1),mesoperioxes,plmeso);
        emfaniseis(thesi,1)=emfaniseis(thesi,1) + 1;
        kentraMazas(thesi,1)=kentraMazas(thesi,1)+ x(j,1);
        xq(j,1)=anamenomeni(thesi,1);
    end

    for j=1:1:levels
        if kentraMazas(j,1)==0
            if j==1
                anamenomeni(j,1)=mesoperioxes(j,1);
            else
                anamenomeni(j,1)=mesoperioxes(j-1,1);
            end
        else
            anamenomeni(j,1)=kentraMazas(j,1)/emfaniseis(j,1);
        end
    end
```

```
    for j=1:1:plmeso
        mesoperioxes(j,1)=(anamenomeni(j,1) + anamenomeni(j+1,1))
            /2;
    end

    D(i,1)=mean((x-xq).^2);
    if abs(D(i,1)-D(i-1,1))<epsilon
        centers=anamenomeni;
        return;
    end
end

function deik= binarysearch(val,mesoperioxes,plmeso)
low=1;
high= plmeso + 1;
deik=fix((low+high)/2);
while(low<high)&&(deik~=1)
    if (val >= mesoperioxes(deik-1,1)) && (val<mesoperioxes(deik,1)
        )
        return;
    elseif val >= mesoperioxes(deik,1)
        low=deik;
    else
        high=deik-1;
    end
    deik=fix((low+high)/2);
    if(high-low)==1
        deik=high;
        return;
    end
end
deik=low;
return;
return;
```

α.

```
SQNRLloydMax=zeros(3,1);
plot_title=['N=2';'N=4';'N=6'];
xq=zeros(length(y),3);
axis_x=[1:1:size(y)];

for i=2:2:6
    [xq(:,i/2),Centers,D]=Lloyd_Max(y,i,-1,1);
```

```
    kmax=length(D);
    loops=[1:1:kmax];
    sqnr_total=zeros(kmax,1);

for j=1:1:kmax
    sqnr_total(j,1)=mean(y.^2)/D(j,1);
end

figure;
plot(loops,sqnr_total,'.-')
title(plot_title(i/2,:));
xlabel('loops');
ylabel('SQNR');
SQNRLloydMax(i/2,1)=sqnr_total(kmax,1);

end
```

β.

```
axis_x=[1:1:size(y)];
my_quantizer_sqnr=zeros(3,1);
plot_title=['Uniform quantization N=2';'Uniform quantization N=4';'
    Uniform quantization N=6'];
for i=2:2:6
    [xq,Centers]=my_quantizer(y,i,-1,1);
    D=mean((y-xq).^2);
    my_quantizer_sqnr(i/2,1)=mean(y.^2)/D;
    figure;
    plot(axis_x,my_quantizer_sqnr(i/2,1),'.');
    title(plot_title(i/2,:));
end
```

γ.

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

D(1,1)=1;
levels=2^N;
lenper=(max_value-min_value)/levels;
mesoperioxes=zeros(levels-1,1);
plmeso=length(mesoperioxes);
epsilon=10^(-16);

for j=1:1:plmeso
    mesoperioxes(j,1)= min_value + j*lenper;
```

```
end

anamenomeni= zeros(levels,1);

for j=1:1:plmeso
    anamenomeni(j,1)=mesoperioxes(j,1)- lenper/2;
end

anamenomeni(levels,1)=mesoperioxes(plmeso,1)+lenper/2;
deigmata = length(x);

for i= 2:1:10^5
    emfaniseis = zeros(levels,1);
    kentraMazas = zeros(levels,1);

    for j=1:1:deigmata
        thesi = binarysearch(x(j,1),mesoperioxes,plmeso);
        emfaniseis(thesi,1)=emfaniseis(thesi,1) + 1;
        kentraMazas(thesi,1)=kentraMazas(thesi,1)+ x(j,1);
        xq(j,1)=anamenomeni(thesi,1);
    end

    for j=1:1:levels
        if kentraMazas(j,1)==0
            if j==1
                anamenomeni(j,1)=mesoperioxes(j,1);
            else
                anamenomeni(j,1)=mesoperioxes(j-1,1);
            end
        else
            anamenomeni(j,1)=kentraMazas(j,1)/emfaniseis(j,1);
        end
    end

    for j=1:1:plmeso
        mesoperioxes(j,1)=(anamenomeni(j,1) + anamenomeni(j+1,1))
            /2;
    end

    D(i,1)=mean((x-xq).^2);
    if abs(D(i,1)-D(i-1,1))<epsilon
        centers=anamenomeni;
        return;
```

```
    end
end

function deik= binarysearch(val,mesoperioxes,plmeso)
low=1;
high= plmeso + 1;
deik=fix((low+high)/2);
while(low<high)&&(deik~=1)
    if (val >= mesoperioxes(deik-1,1)) && (val<mesoperioxes(deik,1)
        )
        return;
    elseif val >= mesoperioxes(deik,1)
        low=deik;
    else
        high=deik-1;
    end
    deik=fix((low+high)/2);
    if(high-low)==1
        deik=high;
        return;
    end
end
deik=low;
return;
return;
```

```
emf_sum = sum(emfaniseis)
p(:)=emfaniseis(:)/emf_sum;
entropy=0;
%To M tha pernei kathe fora timh 2^N (gia N=2 -> M=4)
for i=1:1:M
     entropy = entropy-p(i)*log2(p(i));
end
```

3.3 Ερώτημα 3

```
%duadikh akolouthia:
da=randsrc(100000,1,[0,1]);
```

```
function symbols = gray2(da)
sizeda = length(da);
for i=1:1:sizeda
```

```
    if(da(i)==0)
        symbols(i) = -1;
    elseif(da(i)==1)
        symbols(i) = 1;
    end
end
end
```

```
function symbols = binary2(da)
sizeda = length(da);
for i=1:1:sizeda
    if(da(i)==0)
        symbols(i) = -1;
    elseif(da(i)==1)
        symbols(i) = 1;
    end
end
end
```

```
function symbols = gray4(da)
sizeda = length(da);
m = 1;

for k=1:2:sizeda
    if(da(k)==0 & da(k+1)==0)
        symbols(m) = 0;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1)
        symbols(m) = 2;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0)
        symbols(m) = 3;
         m = m+1;
    end
end
end
```

```
function symbols = binary4(da)
sizeda = length(da);
m = 1;
```

```
for k=1:2:sizeda
    if(da(k)==0 & da(k+1)==0)
        symbols(m) = 0;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0)
        symbols(m) = 2;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1)
        symbols(m) = 3;
         m = m+1;
    end
end
end
```

```
function symbols = gray8(da)
sizeda = length(da);
m = 1;
ypoloipo = mod(sizeda , 3);

if(ypoloipo == 0)

for k=1:3:sizeda
    if(da(k)==0 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = -7;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = -5;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = -3;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = -1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = 3;
```

```
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = 5;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = 7;
         m = m+1;
    end
end
else
   for k=1:3:(sizeda-ypoloipo)
           if(da(k)==0 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = -7;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = -5;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = -3;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = -1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = 3;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = 5;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = 7;
         m = m+1;
    end
   end
end
end
```

```
function symbols = binary8(da)
sizeda = length(da);
m = 1;
```

```
ypoloipo = mod(sizeda , 3);

if(ypoloipo == 0)

for k=1:3:sizeda
    if(da(k)==0 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = -7;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = -5;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = -3;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = -1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = 3;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = 5;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = 7;
         m = m+1;
    end
end
else
   for k=1:3:(sizeda-ypoloipo)
    if(da(k)==0 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = -7;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = -5;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = -3;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1)
```

```
        symbols(m) = -1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1)
        symbols(m) = 3;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0)
        symbols(m) = 5;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1)
        symbols(m) = 7;
         m = m+1;
    end
end
end
end
```

```
function symbols = gray16(da)
sizeda = length(da);
m = 1;

for k=1:4:sizeda
    if(da(k)==0 & da(k+1)==0 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = -5;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = -7;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = -1;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = -3;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = -13;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = -15;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0 & da(k+3)==1)
```

```
        symbols(m) = -9;
         m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = -11;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = 11;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = 9;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = 15;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = 13;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = 3;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = 1;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = 7;
         m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = 5;
         m = m+1;
    end
end
end
```

```
function symbols = binary16(da)
sizeda = length(da);
m = 1;

for k=1:4:sizeda
    if(da(k)==0 & da(k+1)==0 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = -5;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = -7;
```

```
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = -1;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==0 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = -3;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = -11;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = -15;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = -13;
        m = m+1;
    elseif(da(k)==0 & da(k+1)==1 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = -9;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = 5;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = 7;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = 3;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==0 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = 1;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0 & da(k+3)==0)
        symbols(m) = 11;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==0 & da(k+3)==1)
        symbols(m) = 9;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1 & da(k+3)==0)
        symbols(m) = 13;
        m = m+1;
    elseif(da(k)==1 & da(k+1)==1 & da(k+2)==1 & da(k+3)==1)
        symbols(m) = 15;
        m = m+1;
```

```
    end
end
end
```

```
function sm = modulator(symbols,M)

symbolSize = length(symbols);

Tsymbol = 40;
Tc = 4;
fc = 1 / Tc;
Es = 1;

g = sqrt(2 * Es / Tsymbol);

sm = zeros(M, Tsymbol);

for m = 1: symbolSize
    for t = 1: Tsymbol
        sm(m, t) = (2*symbols(m)-1-M) * g * cos(2*pi*fc*t);
    end
end
end
```

```
function [r,N0] = awgn(sm,SNR,M)

Es = 1;
Eb = Es / log2(M);
N0 = Eb/(10^(SNR/10));

[m, n] = size(sm);
noise = sqrt(N0 / 2) * randn(m,n);

r = sm + noise;
```

```
function rnew = demodulator(r)

Tsymbol = 40;

for t = 1: Tsymbol
    y(t, 1) = 1/sqrt(Tsymbol);
end

```

```
rnew = (r * y);
```

```
function symbols1 = foratis(rnew,M)

if M>2
    Eg=1;
else
    Eg=3/M^2-1;
end

for m = 1: M
    s(m, 1) = sqrt(Eg/2)*(2*m-1-M);
end

[rsize, ~] = size(rnew);
for j =1: rsize-1
    for m = 1: M
        tmp(m, 1) = norm([rnew(j,1),rnew(j+1,1)] - s(m,:));
    end
[~, symbols1(j, 1)] = min(tmp);
end

symbols1 = mod(symbols1,M);
```

```
function ektimisi = demapper(symbols1,M,gray)
s1len = length(symbols1);
if M==2
    ektimisi = zeros(s1len,1);
    for k=1:s1len
        if(symbols1(k)==1)
            ektimisi(k)=1;
        elseif(symbols1(k)==(0))
            ektimisi(k)=0;
        end
    end
elseif M==4
    ektimisi = zeros(2*s1len,1);
    if (gray == 0)
        for k=1:s1len
            j = (k*2)-2;
            switch symbols1(k)
                case 0
                    ektimisi(j) = 0;
```

```
                ektimisi(j+1) = 0;
            case 1
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
            case 2
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
            case 3
                ektimisi(j) = 1;
                ektimisi(j+1) = 1;
        end
    end
elseif (gray ==1)
    for k=1:s1len
        j = (k*2)-2;
        switch symbols1(k)
            case 0
                ektimisi(j) = 0;
                ektimisi(j+1) = 0;
            case 1
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
            case 2
                ektimisi(j) = 1;
                ektimisi(j+1) = 1;
            case 3
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
        end
    end
end
elseif M==8
    if (gray == 0)
        ektimisi = zeros(3*s1len,1);
        for k=1:s1len
            j = (k*3)-2;
            switch symbols1(k)
                case -7
                    ektimisi(j) = 0;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 0;
                case -5
                    ektimisi(j) = 0;
```

```
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 1;
            case -3
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 0;
            case -1
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 1;
            case 1
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 0;
            case 3
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 1;
            case 5
                ektimisi(j) = 1;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 0;
            case 7
                ektimisi(j) = 1;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 1;
        end
    end
elseif (gray == 1)
     ektimisi = zeros(3*s1len,1);
     for k=1:s1len
        j = (k*3)-2;
        switch symbols1(k)
            case -7
                ektimisi(j) = 0;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 0;
            case -5
                ektimisi(j) = 0;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 1;
            case -1
                ektimisi(j) = 0;
```

```
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 0;
                case -3
                    ektimisi(j) = 0;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 1;
                case 7
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 0;
                case 5
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 1;
                case 1
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 0;
                case 3
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 1;
            end
         end
    end
elseif M==16
    if (gray == 0)
        ektimisi = zeros(4*s1len,1);
        for k=1:s1len
            j = (k*4)-2;
            switch symbols1(k)
                   case -9
                    ektimisi(j) = 0;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 1;
                    ektimisi(j+3) = 1;
                case -13
                    ektimisi(j) = 0;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 1;
                    ektimisi(j+3) = 0;
                case -11
                    ektimisi(j) = 0;
```

```
            ektimisi(j+1) = 1;
            ektimisi(j+2) = 0;
            ektimisi(j+3) = 0;
        case -15
            ektimisi(j) = 0;
            ektimisi(j+1) = 1;
            ektimisi(j+2) = 0;
            ektimisi(j+3) = 1;
        case -7
            ektimisi(j) = 0;
            ektimisi(j+1) = 0;
            ektimisi(j+2) = 0;
            ektimisi(j+3) = 1;
        case -5
            ektimisi(j) = 0;
            ektimisi(j+1) = 0;
            ektimisi(j+2) = 0;
            ektimisi(j+3) = 0;
        case -1
            ektimisi(j) = 0;
            ektimisi(j+1) = 0;
            ektimisi(j+2) = 1;
            ektimisi(j+3) = 0;
        case -3
            ektimisi(j) = 0;
            ektimisi(j+1) = 0;
            ektimisi(j+2) = 1;
            ektimisi(j+3) = 1;
        case 15
            ektimisi(j) = 1;
            ektimisi(j+1) = 1;
            ektimisi(j+2) = 1;
            ektimisi(j+3) = 1;
        case 13
            ektimisi(j) = 1;
            ektimisi(j+1) = 1;
            ektimisi(j+2) = 1;
            ektimisi(j+3) = 0;
        case 11
            ektimisi(j) = 1;
            ektimisi(j+1) = 1;
            ektimisi(j+2) = 0;
            ektimisi(j+3) = 0;
```

```
            case 9
                ektimisi(j) = 1;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 0;
                ektimisi(j+3) = 1;
            case 7
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 0;
                ektimisi(j+3) = 1;
            case 5
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 0;
                ektimisi(j+3) = 0;
            case 3
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 1;
                ektimisi(j+3) = 0;
            case 1
                ektimisi(j) = 1;
                ektimisi(j+1) = 0;
                ektimisi(j+2) = 1;
                ektimisi(j+3) = 1;
        end
    end
elseif (gray == 1)
    ektimisi = zeros(4*s1len,1);
    for k=1:s1len
        j = (k*4)-2;
        switch symbols1(k)
            case -15
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 1;
                ektimisi(j+3) = 1;
            case -13
                ektimisi(j) = 0;
                ektimisi(j+1) = 1;
                ektimisi(j+2) = 1;
                ektimisi(j+3) = 0;
            case -11
```

```
        ektimisi(j) = 0;
        ektimisi(j+1) = 1;
        ektimisi(j+2) = 0;
        ektimisi(j+3) = 0;
    case -9
        ektimisi(j) = 0;
        ektimisi(j+1) = 1;
        ektimisi(j+2) = 0;
        ektimisi(j+3) = 1;
    case -7
        ektimisi(j) = 0;
        ektimisi(j+1) = 0;
        ektimisi(j+2) = 0;
        ektimisi(j+3) = 1;
    case -5
        ektimisi(j) = 0;
        ektimisi(j+1) = 0;
        ektimisi(j+2) = 0;
        ektimisi(j+3) = 0;
    case -3
        ektimisi(j) = 0;
        ektimisi(j+1) = 0;
        ektimisi(j+2) = 1;
        ektimisi(j+3) = 0;
    case -1
        ektimisi(j) = 0;
        ektimisi(j+1) = 0;
        ektimisi(j+2) = 1;
        ektimisi(j+3) = 1;
    case 15
        ektimisi(j) = 1;
        ektimisi(j+1) = 1;
        ektimisi(j+2) = 1;
        ektimisi(j+3) = 1;
    case 13
        ektimisi(j) = 1;
        ektimisi(j+1) = 1;
        ektimisi(j+2) = 1;
        ektimisi(j+3) = 0;
    case 11
        ektimisi(j) = 1;
        ektimisi(j+1) = 1;
        ektimisi(j+2) = 0;
```

```
                    ektimisi(j+3) = 0;
                case 9
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 1;
                    ektimisi(j+2) = 0;
                    ektimisi(j+3) = 1;
                case 7
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 0;
                    ektimisi(j+3) = 1;
                case 5
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 0;
                    ektimisi(j+3) = 0;
                case 3
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 1;
                    ektimisi(j+3) = 0;
                case 1
                    ektimisi(j) = 1;
                    ektimisi(j+1) = 0;
                    ektimisi(j+2) = 1;
                    ektimisi(j+3) = 1;
            end
         end
    end
end
```