ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

1η Εργασία

Παναγιώτης Κωνσταντίνος Αθανασόπουλος

ΑΜ: 1058112

Έτος 2020-2021

Καθηγητής: Κ.Μπερμπερίδης

Μέρος 1

ΕΡΩΤΗΜΑ 1 – Κωδικοποίηση Huffman

Στα πλαίσια της παρούσας εργασίας πάνω στις Ψηφιακές Τηλεπικοινωνίες μελετήθηκε το πεδίο τής Θεωρίας Κωδικοποίησης Πηγής και ειδικότερα ο Κώδικας Huffman, πού αποτελεί μια τεχνική (αλγόριθμο) κωδικοποίησης τής πληροφορίας μίας διακριτής πηγής.

Με τον όρο «**Κωδικοποίηση Πηγής**» αναφερόμαστε στη διαδικασία πού ακολουθεί μετά την έξοδο μιας πηγής πληροφορίας κι έχει ως πρωταρχικό σκοπό την όσο το δυνατόν αποδοτικότερή αναπαράστασή των δεδομένων πού εξάγει ή πηγή αυτή. Ειδικότερα, στοχεύει στην ελαχιστοποίησή τού μέσου απαιτούμενού μήκους πού απαιτείται για να παρασταθεί κάθε

σύμβολο πού παράγει η πηγή, ως μια κωδική λέξη. Εάν είναι γνωστή ή εντροπία τής πηγής πληροφορίας (μέσος όρος τής πληροφορίας των συμβολών σε bits/σύμβολο), τότε είναι γνωστό και το μέσο πλήθος των bits πού απαιτούνται για να αναπαρασταθούν κατά μέσο όρο τα σύμβολα τής κι εκεί εστιάζουμε την μείωση / συμπίεση.

Ο **«Κώδικας Huffman»** είναι ένας αλγόριθμος για την κωδικοποίησή της πληροφορίας που εξάγει μια πηγή από σταθερό σε μεταβλητό μήκος. Πιο συγκεκριμένα, μπλοκ συμβόλων σταθερού μήκους από την έξοδο τής πηγής απεικονίζονται σε μεταβλητού μήκους μπλοκ δυαδικών συμβόλων. Η βασική αρχή τής μεθόδου αυτής είναι να απεικονίζουμε τις συχνότερα εμφανιζόμενες ακολουθίες σταθερού μήκους σε βραχύτερες δυαδικές ακολουθίες, ενώ οι μακρύτερες δυαδικές ακολουθίες αφήνονται να χρησιμοποιηθούν για τις πιο σπάνια εμφανιζόμενες ακολουθίες τής πηγής. Παρακάτω παρατίθενται τα βήματα υλοποίησής τού αλγορίθμου Human στη γενική περίπτωσή κι ένα τυχαίο παράδειγμα εκτέλεσής τού :

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 σύμβολα «0» και «1»
4. Ταξινόμησε εκ νέου τη λίστα των συμβόλων.
5. Επανέλαβε τα παραπάνω μέχρι όλα τα σύμβολα συγχωνευτούν σε ένα τελικό σύμβολο.

Στο δυαδικό δέντρο πού δημιουργείται :

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

Ανάθεσή Bits σε Σύμβολα Εισόδου :

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

(1) Στο ζητούμενο αύτό καλούμασταν να υλοποιήσουμε στο περιβάλλον τής Matlab τρεις συναρτήσεις πού βασίζονται στην κωδικοποίηση Huffman και να εκτελούν τις παρακάτω λειτουργίες :

- Υπολογισμός των κωδικών λέξεων τής κωδικοποίησής Huffman χρησιμοποιώντας ένα αλφάβητο εισόδου καθώς και τις αντίστοιχες πιθανότητες.
- Συμπίεση / κωδικοποίηση μιας ακολουθίας από σύμβολα σε δυαδικά ψηφία
- Αποσυμπίεση / αποκωδικοποίηση μιας δυαδικής ακολουθίας σε σύμβολα.

α) Το script το οποίο εξωμοιώνει το ερώτημα αυτό είναι το my_hdict.m στο παράρτημα.

β) Το script το οποίο εξωμοιώνει το ερώτημα αυτό είναι το my_henco.m στο παράρτημα.

γ) Το script το οποίο εξωμοιώνει το ερώτημα αυτό είναι το my_hdeco.m στο παράρτημα.

(2) Για την υλοποίηση αυτού τού ζητούμενου, δημιουργήθηκε το script ask1_2.m.Αφού βρούμε το αλφάβητο και προσθέσουμε τις πιθανότητες εμφάνισής των συμβόλων τού για την Πηγή Α κι αφαιρέσουμε περιττά σύμβολα από την αρχική Πηγή Β, εκτελέσαμε τις συναρτήσεις τού 1ου

ζητούμενου για την συμπίεση και αποσυμπιεστή και των δύο πηγών και συγκρίναμε τα τελικά αποτελέσματα τούς με την αρχική πληροφορία πού περιείχε ή κάθε πηγή. Η διαδικασία ολοκληρώθηκε και στις δύο περιπτώσεις επιτυχώς, όπως φαίνεται κι από τα σχετικά μηνύματα πού μας εκτυπωθήκαν στο command window τής Matlab :

```
Huffman decoding of Source A was successful!
Average codeword length of the Huffman code : 4.205070

Huffman decoding of Source B was successful!
Average codeword length of the Huffman code : 4.205070
```

Όπως φαίνεται και παραπάνω, το μέσο μήκος κώδικα είναι παντού ίδιο (και ισούται με 4.205070), γεγονός αναμενόμενο αφού χρησιμοποιήθηκε το ίδιο αλφάβητο και σύνολο πιθανοτήτων για την εξαγωγή τού codebook κάθε κωδικοποίησής Huffman.

Ακόμη, το μήκος τής κωδικοποίησής για την πηγή Α ήταν 42261 δυαδικά ψηφία (bits), ενώ για την πηγή Β το μήκος ανήλθε στα 135482 bits. Παρατηρούμε πώς ή τροποποιημένη Πηγή Β χρειάστηκε εμφανώς μεγαλύτερο πλήθος δυαδικών ψηφίων, όπως αναμενόταν, αφού περιείχε αρκετά μεγαλύτερή ποσότητα πληροφορίας.

(3) Για την υλοποίηση αυτού τού ζητούμενου, δημιουργήθηκε το script κώδικα ask1_3.m. Αφού διαβάσουμε την αρχική (ολόκληρή) Πηγή Β από το αρχείο kwords.txt, βρίσκουμε το ακριβές αλφάβητο τής και υπολογίζουμε τις πιθανότητες εμφάνισής όλων των συμβόλων τού από το αρχείο. Εν συνεχεία, εκτελέσαμε και πάλι τις συναρτήσεις τού 1ου ζητούμενου για την συμπίεση και αποσυμπίεση της Πηγής Β και συγκρίναμε τα τελικά αποτελέσματα με την αρχική πληροφορία πού περιείχε η πηγή για να επιβεβαιώσουμε την επιτυχία τής διαδικασίας. Όπως φαίνεται κι από το μήνυμα πού μας επιστράφηκε κατά την ολοκλήρωση τής εκτέλεσής, όλα τελείωσαν επιτυχώς :

```
Huffman decoding of Source B was successful!
Average codeword length of the Huffman code : 4.132011
```

Στήν περίπτωσή αυτή το μήκος κώδικοποίήσής τής Πήγής Β βλέπούμε πώς ισούται με 121415 δύαδικά ψήφία, τιμή πού είναι προφανώς μικρότερή από το μήκος κωδικοποίησής τής τροποποιημένης Πηγής Β τού προηγούμενού ζητούμενου (= 135482 bits). Αυτό οφείλεται στο γεγονός πώς οι πιθανότητες εμφάνισής των συμβόλων και το αλφάβητο πού χρησιμοποιήθηκε, είναι υπολογισμένα ακριβώς για τη συγκεκριμένη πηγή, και δεν αποτελούν κάποια γενική θεώρηση πιθανοτήτων για τα πεζά γράμματα τού αγγλικού αλφαβήτου όπως νωρίτερα. Αυτή είναι και ή αιτία πού και το μέσο μήκος κώδικα είναι τώρα μικρότερο. Όλα τα παραπάνω μπορούν να «μεταφραστούν» ως μια αποδοτικότερή κωδικοποίηση πηγής και καλύτερή συμπίεση τής πληροφορίας τής Πηγής Β.

(4) Για την υλοποίηση αυτού τού ζητούμενου, δημιουργήθηκε ο κώδικας τού αρχείου ask1_4.m. Ειδικότερα, αφού βρούμε το αρχικό αλφάβητο και προσθέσουμε τις πιθανότητες εμφάνισής των συμβόλων τού για την Πηγή Α σύμφωνα με τα δεδομένα από το Ζητούμενο 2, δημιουργούμε την δεύτερής τάξης επέκτασή τής Πηγής Α και υπολογίζουμε το νέο αλφάβητο τής Πηγής και τις αντίστοιχες πιθανότητες εμφάνισης των συμβόλων. Κατόπιν, εκτελούμε τις συναρτήσεις τού 1ου ζητούμενου για την συμπίεση και αποσυμπίεση και συγκρίνουμε τα τελικά αποτελέσματα τούς με την αρχική πληροφορία πού περιείχε η πηγή μας. Το μήνυμα πού μας επιστρέφεται κατά την ολοκλήρωση όλης τής εκτέλεσής είναι :

```
Huffman decoding of Source A was successful!
Average codeword length of the Huffman code : 8.381934
```

Όπως παρατηρούμε από το παραπάνω μήνυμα, η κωδικοποίησή και αποκωδικοποίηση ολοκληρωθήκαν επιτυχώς, ενώ το μέσο μήκος κώδικα ισούται με 8.381934 δυαδικά ψηφία και το μήκος τής κωδικοποιημένης πηγής, πού αποτελεί επέκτασή 2ης τάξης τής αρχικής πηγής Α, είναι 41960 bits.

Συγκρίνοντας τα αποτελέσματα μας με αυτά τού 2ου ζητούμενου, φαίνεται πώς η κωδικοποίησή – συμπίεση τής «νέας» Πηγής Α είναι καλύτερή από την αρχική, αφού το μήκος τής κωδικοποιημένης πηγής είναι μικρότερο κι επομένως βέλτιστο σε σχέση με το αρχικό (41960 < 42261).

Ο παραπάνω ισχυρισμός επιβεβαιώνεται και με βάση τη θεωρία για την επέκτασή τής τάξης μιας πηγής, σύμφωνα με την οποία γενικότερα ή ν – οστή επέκτασή πηγής αποφέρει κώδικες πού είναι ολοένα και πιο κοντά στο όριο μέγιστής συμπίεσής, δηλαδή την τιμή τής εντροπίας, τής πηγής.

Τέλος, να σημειώσουμε ότι στην περίπτωση τής 2ης τάξης επέκτασής πηγής, η αποσυμπίεση είναι αρκετά πιο χρονοβόρα σε σχέση με προηγούμενα ζητούμενα, κι αποτελεί μεγαλύτερή υπολογιστική ισχύ.

(5) Για την υλοποίηση αυτού του ζητούμενου, δημιουργήθηκε το script κώδικα ask1_5.m. Αρχικά, πρέπει να τροποιήσουμε και να κωδικοποιήσουμε την Πηγή Β σύμφωνα με την 2ης τάξης επέκτασή τού αλφαβήτου της Πηγής Α και τις αντίστοιχες πιθανότητες για τα ζεύγη χαρακτήρων, πού υπολογίστηκαν και στο προηγούμενο ζητούμενο. Έτσι, υπολογίζουμε ομοίως με παραπάνω το νέο αλφάβητο και τις νέες πιθανότητες της επεκταμένης πηγής Α και τροποποιούμε την Πηγή Β ώστε να μπορεί να κωδικοποιηθεί με βάση το παραπάνω αλφάβητο, αφαιρώντας / τροποιώντας τα επιπλέον σύμβολα. Έπειτα, χωρίζουμε την τροποποιημένη Πηγή Β σε ζεύγη χαρακτήρων και υλοποιούμε την συμπίεση – αποσυμπίεστη Huffman με χρήση των συναρτήσεων τού 1ου ζητούμενου. Τέλος συγκρίνουμε το τελικό αποτέλεσμα με την αρχική πληροφορία και μας επιστρέφεται σχετικό μήνυμα, πού φαίνεται παρακάτω :

```
Huffman decoding of Source B was successful!
Average codeword length of the Huffman code : 8.381934
```

Όπως είναι προφανές το 1ο μέρος τής συνολικής διαδικασίας ολοκληρώνεται επιτυχώς, ενώ το μέσο μήκος κώδικα ισούται με 8.381934 (ίδιο με τού προηγούμενου ζητουμένου, αφού χρησιμοποιήσαμε ίδιο αλφάβητο, σύνολο πιθανοτήτων) και το μήκος τής κωδικοποιημένης πηγής ανέρχεται σε 134429 δυαδικά ψηφία. Και εδώ, όπως ορίζει και ή θεωρία για την επέκτασή πηγής, το μήκος κωδικοποίησης είναι μικρότερο από το αρχικό βάση τού αλφαβήτου της Πηγής Α (134429 < 135482) κι επομένως η συμπίεση είναι πιο αποδοτική, ενώ απαιτείται σημαντικά μεγαλύτερος χρόνος για την ολοκλήρωση της αποσυμπίεσης.

Στη συνέχεια, πρέπει να δημιουργήσουμε τής 2ης τάξης επέκτασή τής αρχικής Πηγής Β, κι αφού βρούμε το νέο αλφάβητο και εκτιμήσουμε τις νέες πιθανότητες εμφάνισής των ζευγών συμβόλων συγκεκριμένα για την πηγή αυτή, όπως στο ζητούμενο 3, να εκτελέσουμε τις σχετικές συναρτήσεις για την συμπίεση κι αποσυμπίεση Huffman της επεκταμένης Πηγής Β.

ΕΡΩΤΗΜΑ 2 – Κωδικοποίηση PCM

Το περιεχόμενο της εργαστηριακής άσκησης είναι η μέλετη βασικών μεθόδων για το σχεδιασμό συστημάτων κωδικοποίησης κυματομορφής. Τα συστήματα αυτά σχεδιάζονται ώστε να επιτρέπουν την αναπαραγωγή στον προορισμό της κυματομορφής εξόδου της πηγής με όσο τον δυνατό μικρότερη παραμόρφωση. Η παλμοκωδική διαμόρφωση (PCM), που είναι και το αντικείμενο της άσκησης, είναι ένα σύστημα ψηφιακής διαμόρφωσης αναλογικών δεδομένων.

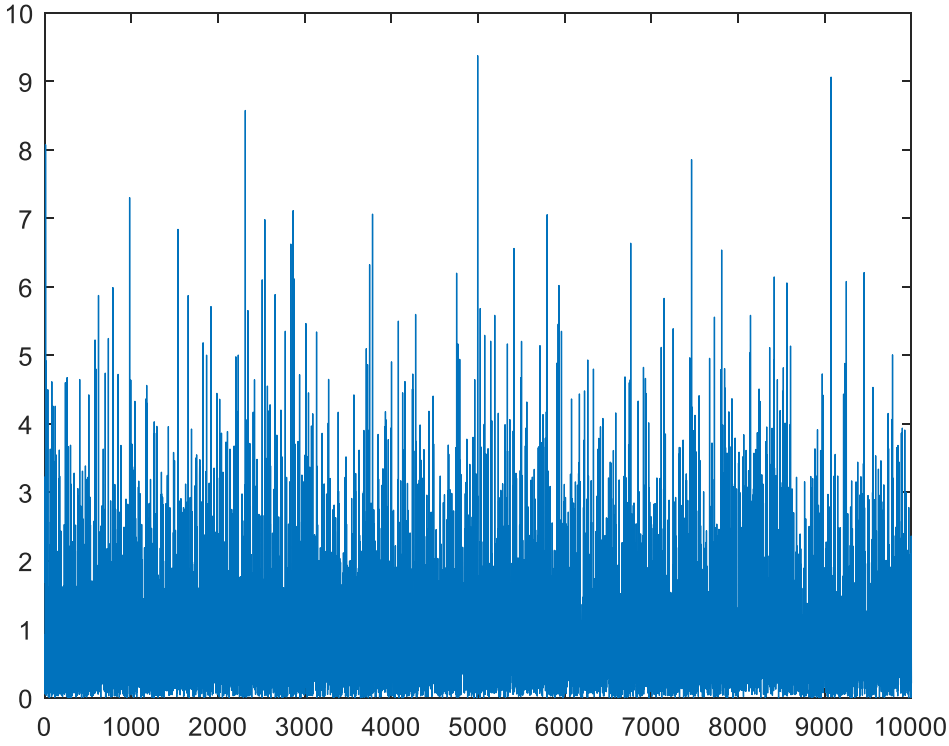
Ειδικότερα ασχοληθήκαμε με την κατασκευή του κβαντιστή του PCM συστήματος. Φτιάξαμε συναρτήσεις τόσο για ομοιόμορφο κβαντιστή (ομοιόμορφο PCM) όσο και για μη ομοιόμορφο κβαντιστή (μη ομοιόμορφο PCM). Μελετήσαμε επίσης και την περίπτωση του PCM που χρησιμοποιεί μη ομοιόμορφο κβαντιστή, τον οποίο σχεδιάσαμε υλοποιώντας τον αλγόριθμο Lloyd-Max.

Η PCM είναι μια μέθοδος κωδικοποίησης κυματομορφής, η οποία μετατρέπει ένα αναλογικό σήμα σε ψηφιακά δεδομένα. Έχει τρία βασικά μέρη: Δειγματολήπτη, κβαντιστή και κωδικοποιητή. Για την άσκηση υλοποιήθηκε ένας μη ομοιόμορφος κβαντιστής N bits , δηλαδή 2^N επιπέδων.

2) Δημιουργώ την πηγή Α με βάση την εκφώνηση:

```
t = (randn(10000,1)+j*randn(10000,1))/sqrt(2);

x= abs(t) .^ 2;
```

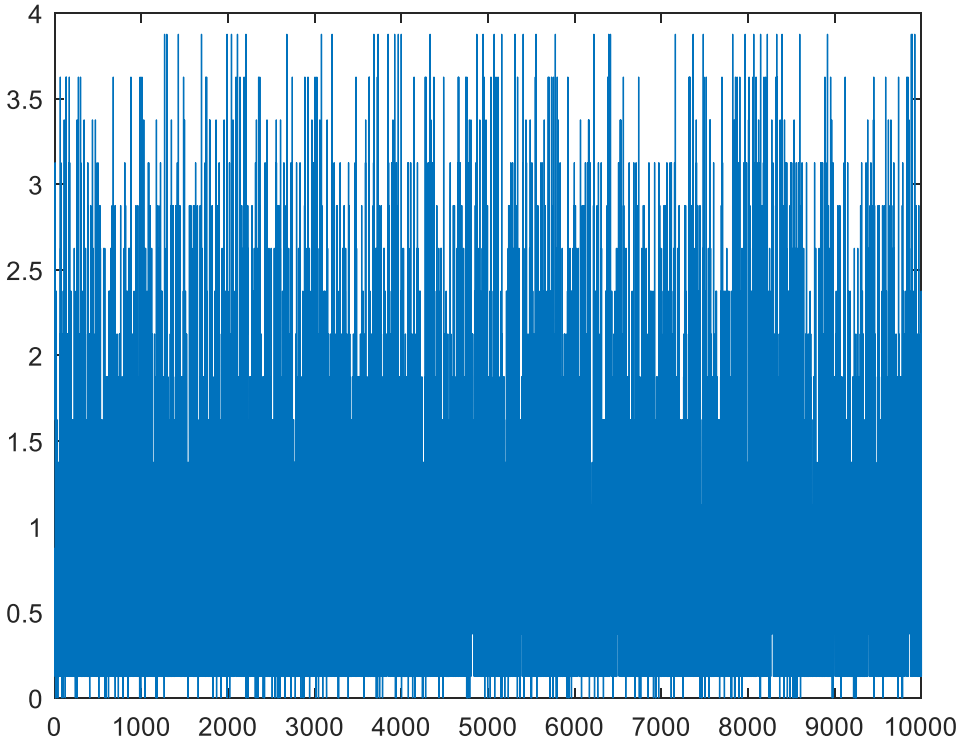
Το αρχικό σήμα φαίνεται παρακάτω:



Στη συνέχεια χρησιμοποιούμε τον ομοιόμορφο κβαντιστή που υλοποιήσαμε για να κωδικοποιήσουμε την πηγή A με min_value = 0, max_value = 4 και έχω:

Για N = 4 bits

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

plot(xq);
```



a) Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή και την θεωρητική παραμόρφωση:

[my4_sqnr_exp, my4_sqnr_theor] = sqnr(x, xq, 2);

Το οποίο μας επιστρέφει 6.3567 και 13.799 αντίστοιχα.

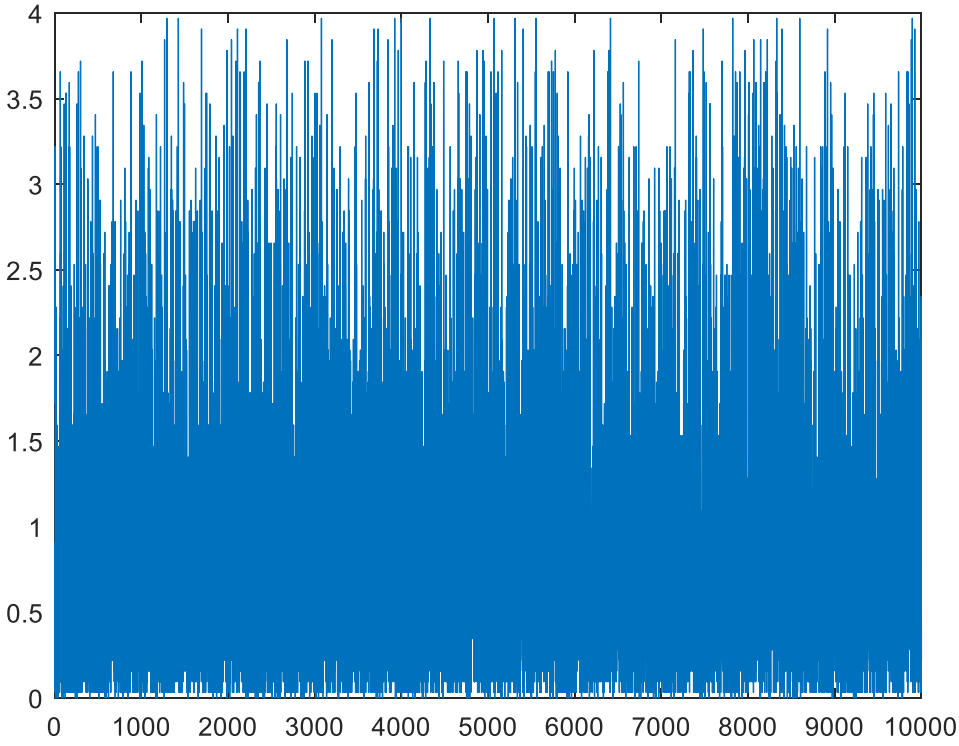
b) Η πιθανότητα να βρεθεί η είσοδος του κβαντιστή εκτός δυναμικής περιοχής είναι:

```
>> disp(p);
    0.2259    0.1733    0.1347    0.1084    0.0780    0.0594    0.0543    0.0376    0.0277    0.0223    0.0174    0.0145    0.0117    0.0062    0.0075    0.0044
```

Για N = 6 bits

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

plot(xq);



a) Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή και την θεωρητική παραμόρφωση:

[my4_sqnr_exp, my4_sqnr_theor] = sqnr(x, xq, 2);

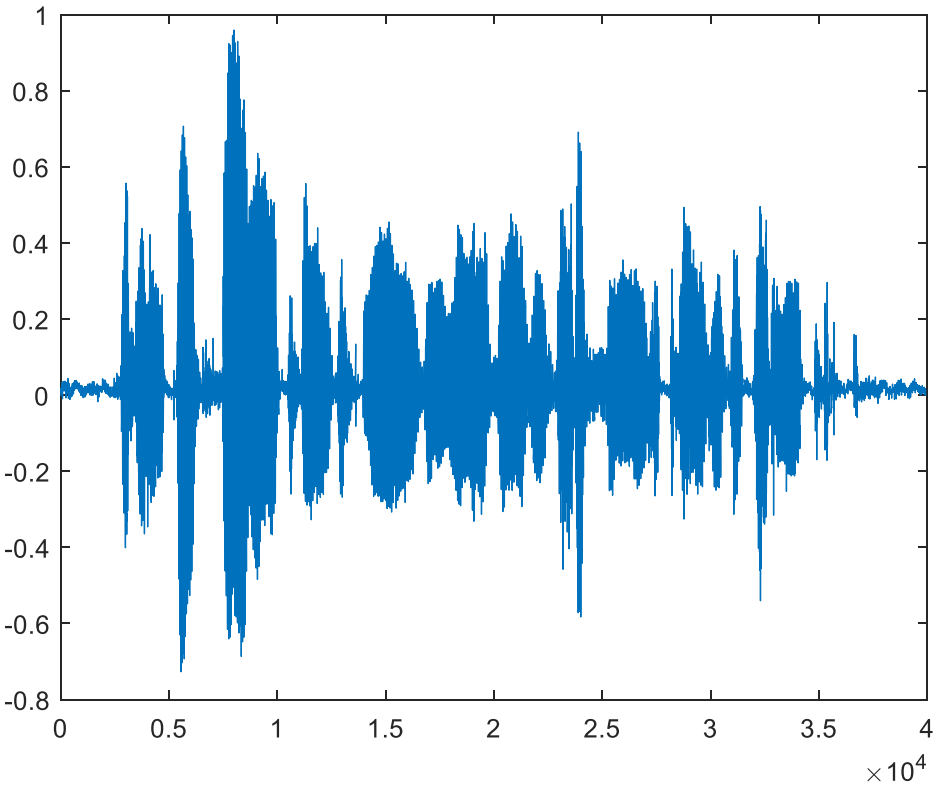
Το οποίο μας επιστρέφει 6.4038 και 13.799 αντίστοιχα.

b) Η πιθανότητα να βρεθεί η είσοδος του κβαντιστή εκτός δυναμικής περιοχής είναι:

```
>> disp(p);
  Columns 1 through 16

    0.0635    0.0537    0.0567    0.0520    0.0462    0.0436    0.0420    0.0415    0.0365    0.0355    0.0336    0.0291    0.0277    0.0284    0.0286    0.0237

  Columns 17 through 32

    0.0222    0.0202    0.0193    0.0163    0.0175    0.0155    0.0124    0.0140    0.0173    0.0123    0.0125    0.0122    0.0089    0.0093    0.0104    0.0090

  Columns 33 through 48

    0.0071    0.0075    0.0065    0.0066    0.0056    0.0057    0.0057    0.0053    0.0051    0.0046    0.0035    0.0042    0.0034    0.0044    0.0030    0.0037

  Columns 49 through 64

    0.0020    0.0029    0.0027    0.0041    0.0019    0.0012    0.0016    0.0015    0.0018    0.0016    0.0026    0.0015    0.0010    0.0010    0.0010    0.0014
```

1) [y,fs,N]=wavread('speech.wav');

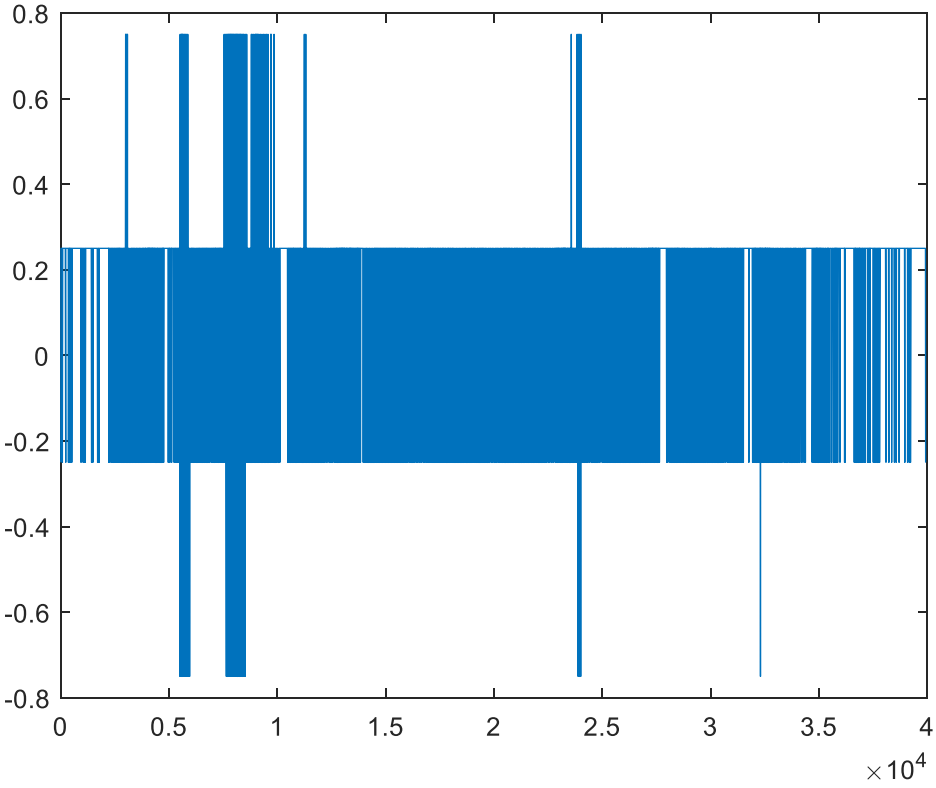
Το αρχικό σήμα της πηγής φαίνεται παρακάτω:



Αρχικά, χρησιμοποιόντας το αρχείο my_quantizer.m για τον ομοιόμορφο κβαντιστή κωδικοποιούμε την πηγή Β για min_value = -1, max_value = 1:

Για N = 2 bits

```
[xq, centers, p] = my_quantizer(y,2,-1,1);

plot(xq);
```



Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

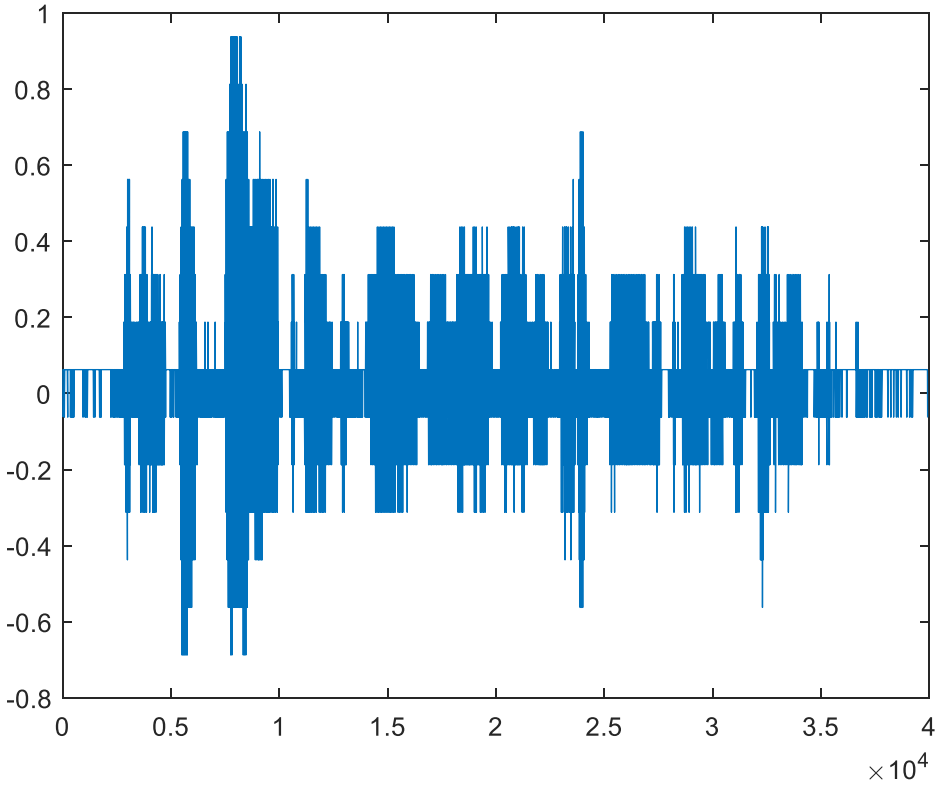
```
sqnr2my = sqnr(y,xq,2);
```

Το οποίο μας επιστρέφει -2.90006.

Για N = 4 bits

```
[xq, centers, p] = my_quantizer(y,4,-1,1);
```

```
plot(xq);
```



Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

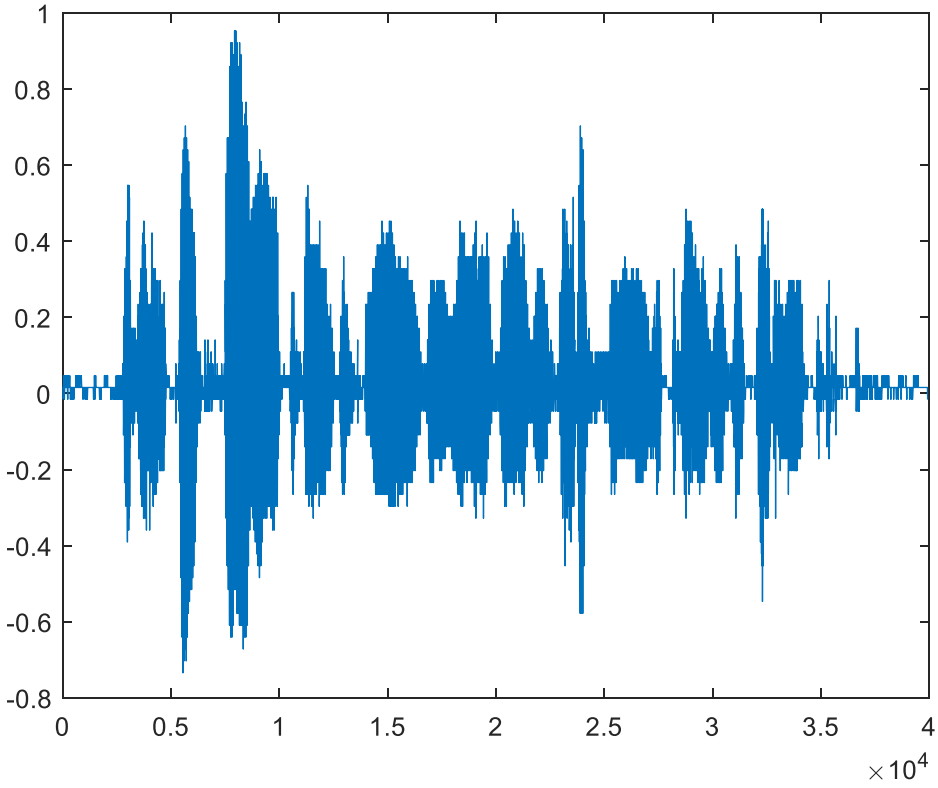
```
sqnr4my = sqnr(y,xq,4);
```

Το οποίο μας επιστρέφει 10.7555.

Για N = 6 bits

[xq, centers, p] = my_quantizer(y,6,-1,1);

plot(xq);



Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

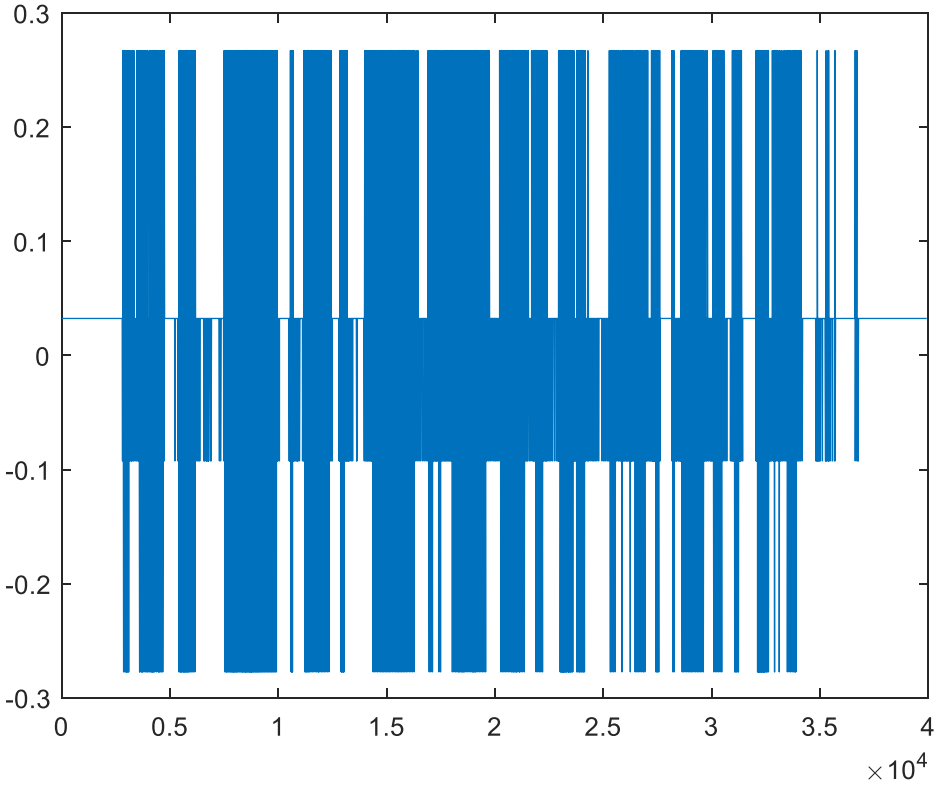
```
sqnr6my = sqnr(y,xq,6);
```

Το οποίο μας επιστρέφει 23.705.

Στη συνέχεια, χρησιμοποιώντας το αρχείο Lloyd_Max.m για τον αλγόριθμο Lloyd-Max κωδικοποιούμε την πηγή B για min_value = -1, max_value = 1:

Για N = 2 bits

```
[xq, centers, D, p] = Lloyd_Max(y,2,-1,1);
plot(xq);
```



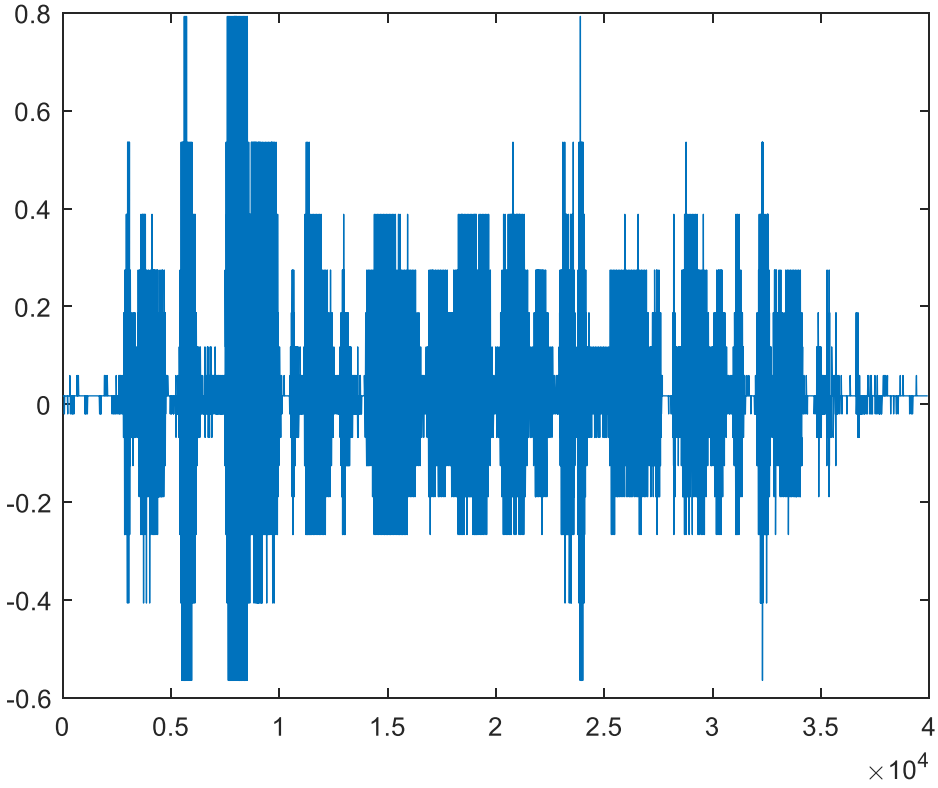
Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

```
sqnr2lloyd = sqnr(y,xq,2);
```

Το οποίο μας επιστρέφει 7.1945.

Για N = 4 bits

```
[xq, centers, D, p] = Lloyd_Max(y,4,-1,1);
plot(xq);
```



Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

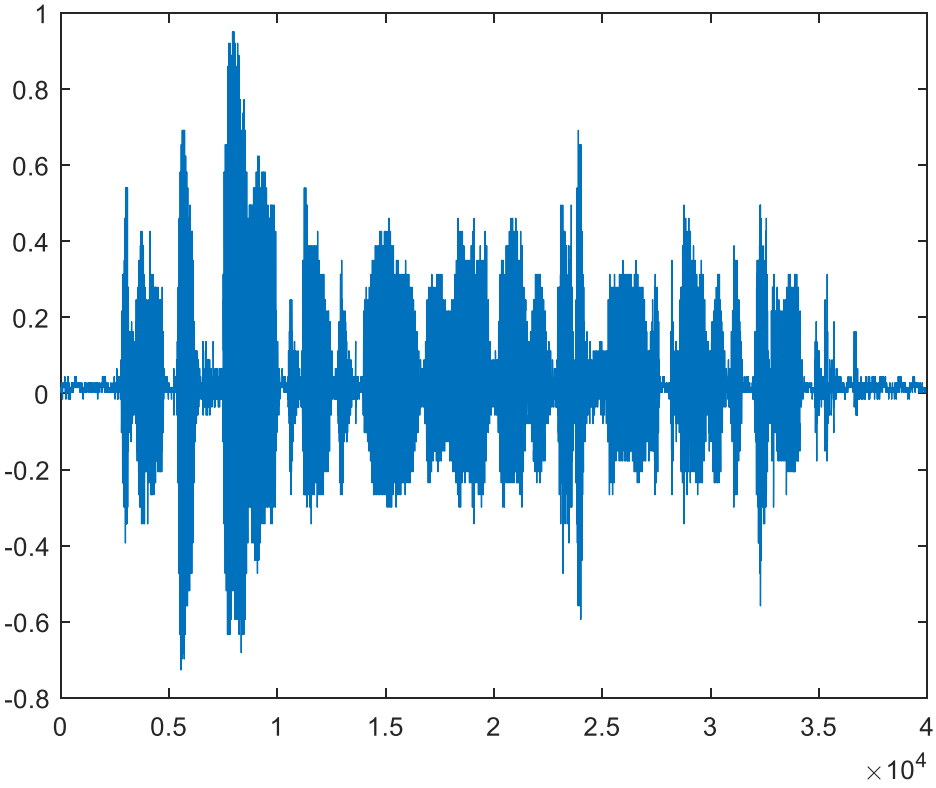
```
sqnr4lloyd = sqnr(y,xq,4);
```

Το οποίο μας επιστρέφει 17.7441.

Για N = 6 bits

```
[xq, centers, D, p] = Lloyd_Max(y,6,-1,1);
```

```
plot(xq);
```



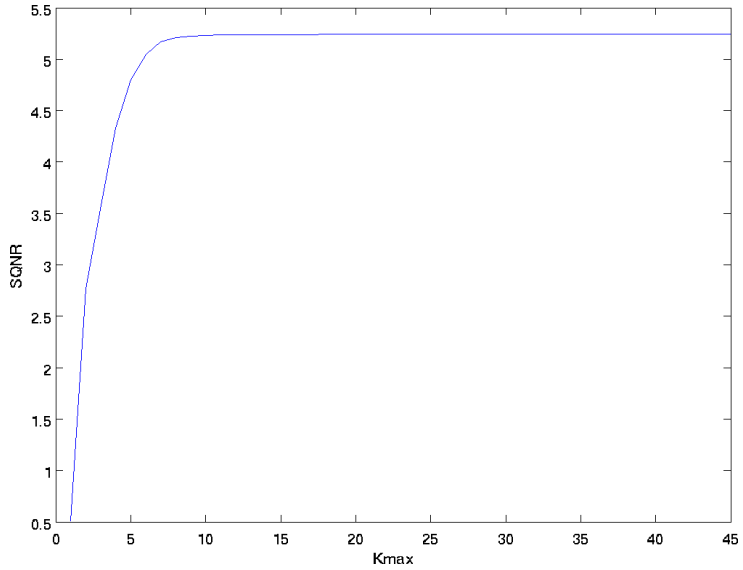
Στη συνέχεια με τη συνάρτηση sqnr() υπολογίζουμε το SQNR στην έξοδο του κβαντιστή:

```
sqnr6lloyd = sqnr(y,xq,6);
```

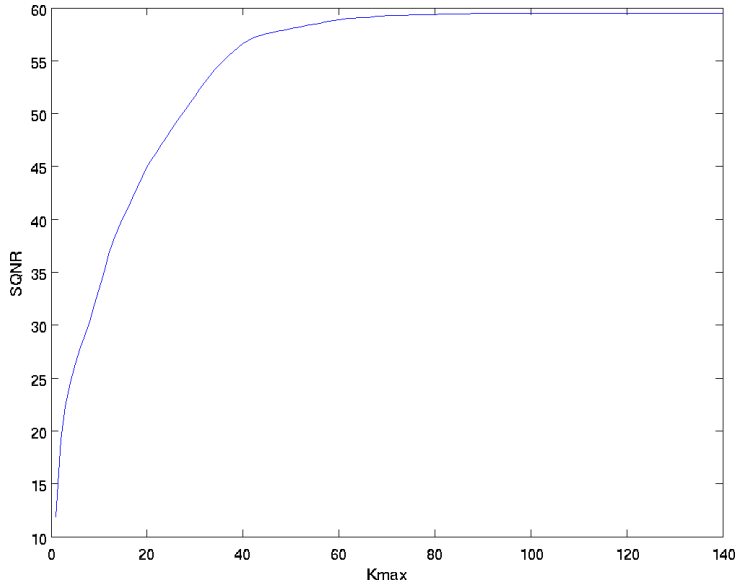
Το οποίο μας επιστρέφει 26.76.

a.

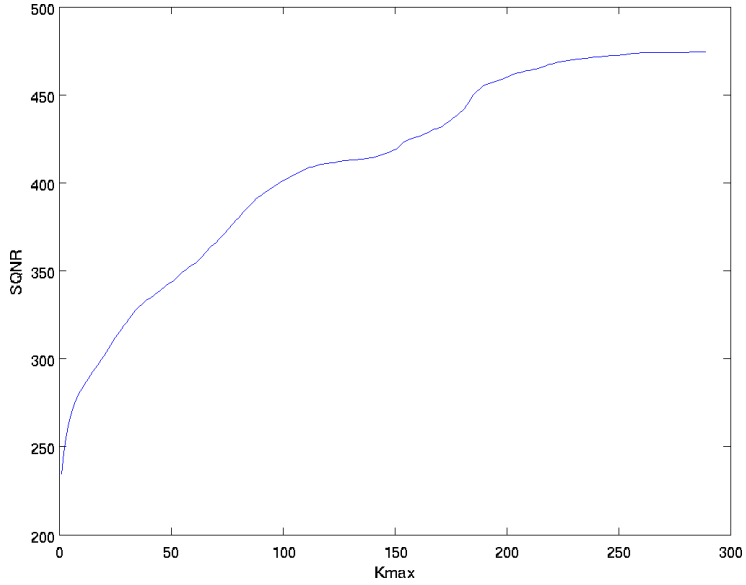
Ο ρυθμός με τον οποίο μεταβάλλεται το SQNR σε σχέση με τον αριθμό των επαναλήψεων:



$N = 2$



$N = 4$



$N = 6$

b.

Ο υπολογισμός των SQNR υλοποιείται με το script erotima2b.m

Αριθμός N bits	SQNR ομοιόμορφου κβαντιστή	SQNR Lloyd Max Κβαντιστή
2	-2.9001	7.1946
4	10.7555	17.7441
6	23.705	26.76

c.

Για να υπολογίσουμε πειραματικά την πιθανότητα εμφάνισης κάθε στάθμης του κβαντιστή αρκεί να υπολογίσουμε την πιθανότητα εμφάνισης του κάθε κέντρου κβάντισης. Η υλοποίηση γίνεται με

την συνάρτηση exp_freq()και το script erotima2c.m

για N = 2:

```
>> disp(freq2bits_exp) 0.0594
    0.1943
    0.6321
```

```
    0.1142
```

και επαληθεύω:

```
>> sum(freq2bits_exp)

ans =

    1.0000
```

- για N = 4:

```
>> disp(freq4bits_exp) 0.0036
    0.0076
    0.0240
    0.0452
    0.0627
    0.0790
    0.1205
    0.3549
    0.1106
    0.0801
    0.0538
    0.0352
    0.0147
    0.0065
    0.0015
```

και επαληθεύω:

```
>> sum(freq4bits_exp)

ans =

     1
```

- και τέλος για N = 6:

```
>> disp(freq6bits_exp) 0.0000
    0.0001
    0.0002
```

0.0005
0.0006
0.0007
0.0013
0.0015
0.0019
0.0026
0.0032
0.0052
0.0084
0.0123
0.0165
0.0197
0.0222
0.0256
0.0269
0.0309
0.0360
0.0481
0.0553
0.1072
0.1542
0.1085
0.0586
0.0433
0.0356
0.0316
0.0274
0.0218
0.0201
0.0148
0.0140
0.0115
0.0093
0.0057
0.0041
0.0036
0.0025
0.0020
0.0015

0.0009

0.0008

```
    0.0004
    0.0003
    0.0002
    0.0002
    0.0001
    0.0001
    0.0002
    0.0001
    0.0002
    0.0001
```

και επαληθεύω:

```
>> sum(freq6bits_exp)


ans =


     1
```

Για τον υπολογισμό της εντροπίας από την θεωρία γνωρίζουμε οτι ο τύπος με τον οποίο υπολογίζεται είναι:

$$H = \sum p_i * \log_2 * (1/p_i)$$

όπου p είναι η πιθανότητα εμφάνησης κάθε στάθμης.

Για τον υπολογισμό της εντροπίας χρησιμοποιώ την συνάρτηση entropy()και τα αποτελέσματα που λαμβάνω είναι:

- για N = 2 έχω:

```
entropy2 = entropy(xq2lloyd);
>> disp(entropy2)
    1.4771
```

- για N = 4 έχω:

```
>> entropy4 = entropy(xq4lloyd);
>> disp(entropy4)
    3.0429
```

- για N = 6 έχω:

```
>> entropy6 = entropy(xq6lloyd);

>> disp(entropy6)

    4.4235
```

d.

Υπολογίζοντας το τετραγωνικό μέσο σφάλμα (Mean Square Error – MSE) πού προκύπτει για κάθε περίπτωση κωδικοποίησης PCM χρησιμοποιώντας τόσο τον Ομοιόμορφο, όσο και τον Μη Ομοιόμορφο Κβαντιστή για το στάδιο ths κβάντισης λαμβάνουμε τα εξής αποτελέσματα :

```
The Mean Square Error of Lloyd_Max, Source B, for
N=2 is: 0.003237 The Mean Square Error of
my_quantizer, Source B,for N=2 is: 0.144851

The Mean Square Error of Lloyd_Max, Source B, for
N=4 is: 0.000270 The Mean Square Error of
my_quantizer, Source B,for N=4 is: 0.113370

The Mean Square Error of Lloyd_Max, Source B, for
N=6 is: 0.000034 The Mean Square Error of
my_quantizer, Source B,for N=6 is: 0.113241
```

Όπως είναι προφανές από τα παραπάνω, η τιμή τού σφάλματος (MSE) πού παρουσιάζεται στην πιστότητα τού κωδικοποιημένου σήματος σε σχέση με το αρχικό δοθέν, μειώνεται σημαντικά καθώς αυξάνεται το πλήθος των δυαδικών ψηφίων πού διαθέτουμε για την κωδικοποίηση της εισόδου. Αυτό πρακτικά μπορεί να ερμηνευτεί στην περίπτωση τής δικής μας πηγής ως λιγότερή παραμόρφωση στην κυματομορφή, και συνεπώς πιο καθαρό ήχο στο τελικό αποτέλεσμα πού λαμβάνουμε ως έξοδο.

Ακόμη, παρατηρούμε πώς ένας άλλος παράγοντας πού συντελεί στην μείωση τού σφάλματος εξόδου, και κατ' επέκτασή σε μεγαλύτερή αποδοτικότητα τής κωδικοποίησης PCM είναι η χρήση Μη Ομοιόμορφού Κβαντιστή για το στάδιο τής κβάντισης τής εισόδου, αφού το MSE παραμένει σε κάθε περίπτωση (N=2,4,6) μικρότερο από την αντίστοιχή τιμή με χρήση Ομοιόμορφού Κβαντιστή.

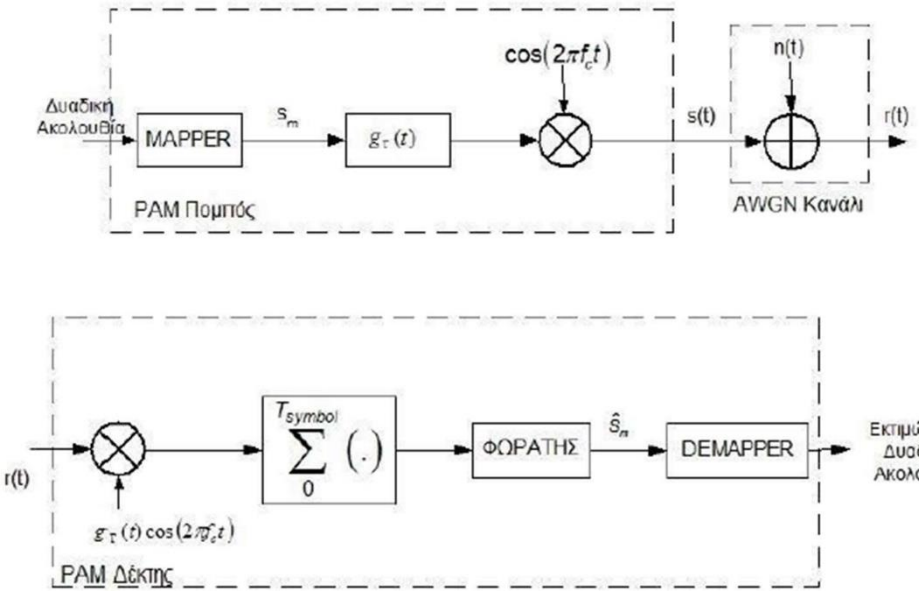
Συνοψίζοντας τα άνωθεν αναφερθέντα, ή κωδικοποίηση κυματομορφών PCM είναι μια αρκετά αποδοτική μέθοδος μετατροπής αναλογικών δεδομένών σε ψηφιακά, τής οποίας ή αποδοτικότητα εξαρτάται πολύ από το πλήθος δυαδικών ψηφίων κωδικοποίησης και το είδος του κβαντιστή. Για τη βελτιστοποίηση τής τελικής εξόδου συνιστάται η χρήση όσο το δυνατόν μεγαλύτερού πλήθους δυαδικών ψηφίων για την κωδικοποίηση, και Μη Ομοιόμορφος Κβαντιστής για την κβάντιση.

ΕΡΩΤΗΜΑ 3 – Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-PAM

ΖΗΤΟΥΜΕΝΟ 1

Εδώ έπρεπε να υλοποιηθεί προσομοίωση τηλεπικοινωνιακού συστήματος M-PAM ώστε να μετρηθεί η απόδοση για M=2,4,8 και M=16.

Σε ένα σύστημα M-PAM ο πομπός δέχεται ως είσοδο μια δυαδική ακολουθία, τη μετατρέπει σε σύμβολα, την πολλαπλασιάζει με τον ορθογώνιο παλμό, και κατόπιν το σήμα μεταφέρεται στη ζώνη μετάδοσης μέσω του διαμορφωτή. Στο σήμα που στάλθηκε προστίθεται AWGN θόρυβος, και φθάνει στο δέκτη του M-PAM συστήματος. Εκεί αποδιαμορφώνεται και προκύπτει ένα μονοδιάστατο διάνυσμα το οποίο εισάγεται στο φωρατή όπου και αποφασίζεται ποιο σύμβολο στάλθηκε. Τέλος, ο demapper κάνει την αντίστροφη αντιστοίχιση από σύμβολα σε bits.



Σαν είσοδο στο σύστημα δίνω μια ακολουθία εισόδου των 10^5 bits η οποία δημιουργείται τυχαία μέσω της συνάρτησης randsrc.

Ο mapper είναι ένας μετατροπέας από bits σε σύμβολα. Κάθε σύμβολο αντιστοιχεί σε μια συγκεκριμένη ακολουθία $\log_2 M$ bits. Δηλαδή ο mapper για κάθε log2M bits εξάγει ένα από τα σύμβολα της διαμόρφωσης M-PAM. Αντίστοιχα ο demapper δέχεται ως είσοδο σύμβολο και εξάγει μια ακολουθία log2M bits για κάθε σύμβολο.

Στην κωδικοποίηση Gray αν δυο σύμβολα είναι γειτονικά τότε σε αυτά ανατίθενται διατάξεις bits που διαφέρουν μόνο κατά 1 bit μεταξύ τους.

Η μετάδοση γίνεται μέσω ορθογώνιου παλμού τον οποίο προσομοιώνουμε. Για την δημιουργία του ορθογώνιου παλμού χρησιμοποιείται στο modulation ο τύπος g_T που έχει δοθεί στην εκφώνηση.

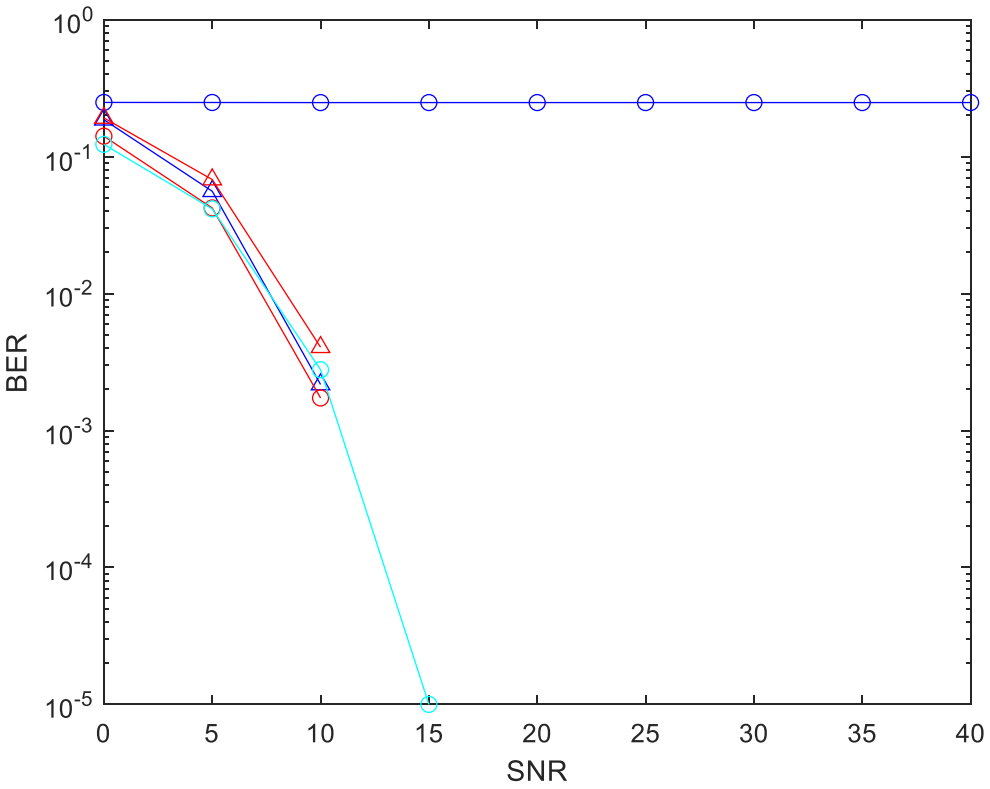
Το ζωνοπερατό σήμα που εκπέμπει ο πομπός διέρχεται μέσα από ένα ιδανικό κανάλι προσθετικού θορύβου. Ο θόρυβος είναι λευκός και ακολουθεί Gaussian κατανομή μέσης τιμής. Τον παράγω με κλήση της συνάρτησης randn.

Ο αποδιαμορφωτής του συστήματος M-PAM συσχετίζει το ληφθέν σήμα με τη φέρουσα και τον ορθογώνιο παλμό. Η συσχέτιση γίνεται στα χρονικά πλαίσια μιας περιόδου συμβόλου. Ο αποδιαμορφωτής συσχετίζει το σήμα με την συνιστώσα της φέρουσας οπότε προκύπτει το διάνυσμα r που είναι η εκτιμηθείσα τιμή του τρέχοντος συμβόλου.

Ο φωρατής με βάση την τιμή r αποφασίζει σε ποιο σύμβολο βρίσκεται πιο κοντά. Το σύμβολο που θα έχει την μικρότερη απόσταση από το r είναι τελικά το σύμβολο που στάλθηκε.

ΖΗΤΟΥΜΕΝΟ 2

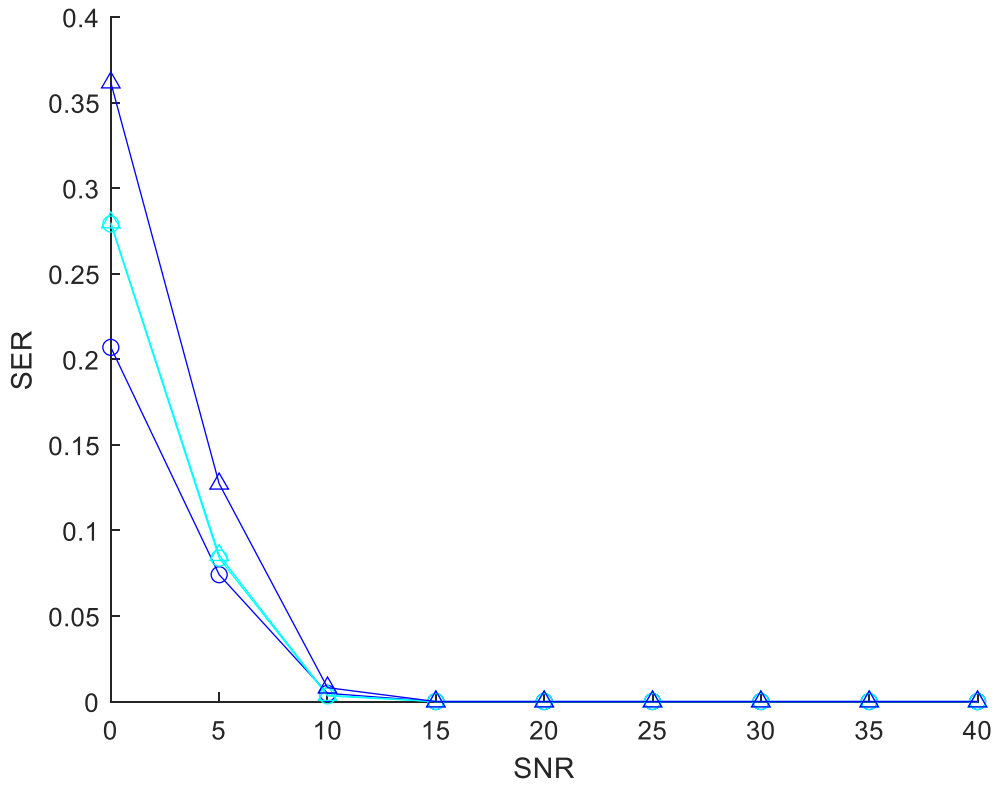
Οι καμπύλες που προκύπτουν παρουσιάζονται σε λογαριθμική κλίμακα και έχουν μεγάλη ομοιότητα με τις θεωρητικές.



Είναι φανερό ότι όσο μεγαλώνει ο αριθμός Μ, τόσο μειώνεται και η ανοχή στα σφάλματα, αλλά αυξάνεται ο ρυθμός μετάδοσης.

ΖΗΤΟΥΜΕΝΟ 3

Οι καμπύλες που προκύπτουν παρουσιάζονται σε λογαριθμική κλίμακα και έχουν μεγάλη ομοιότητα με τις θεωρητικές.



Από το διάγραμμα αυτό προκύπτουν τα εξής συμπεράσματα. Όπως παρατηρείται, όσο μεγαλώνει ο αριθμός M, τόσο μειώνεται και η ανοχή στα σφάλματα, αλλά αυξάνεται ο ρυθμός μετάδοσης. Επίσης, όπως είναι αναμενόμενο, και για τις δυο τιμές του M, η κωδικοποίηση κατά Gray είναι πιο αποδοτική καθώς δίνει μικρότερα σφάλματα από την αντίστοιχη κανονική κωδικοποίηση.

Στην υλοποίηση μου χρησιμοποιώ τις συναρτήσεις mainb, myresult , mapper και demapper .

ΠΑΡΑΡΤΗΜΑ ΜΕ ΚΩΔΙΚΕΣ

Στο σημείο αυτό παρουσιάζουμε τους κώδικες που υλοποιήσαμε για κάθε ένα ερώτημα αλλά και διάφορα script αρχεία για έλεγχο.

my_hdict.m

```
function [dict,avglen] = my_hdict(abet, prob)
% Erotima 1/1/(a) Code

% Preallocation
ROUND_OFF_ERROR = 1e-6;
```

```
n_ary = 2;
variance = 'max';

% Check if any of the elements of probability vector is negative
if length(find(prob < 0))~=0,
    disp('There are negative values in the probability vector!')
end

% Check if the sum of the elements of probability vector is equal to 1
if abs(sum(prob)-1)>10e-10,
    disp('The probability vector elements do not add up to 1!')
end


% Create tree nodes with the signals and the corresponding probabilities
huff_tree = struct('signal', [], 'probability', [], 'child', [], 'code', [],
'origOrder', -1);

for i=1:length(abet)
    huff_tree(i).signal = abet{i};
    huff_tree(i).probability = prob(i);
    huff_tree(i).origOrder = i;
end


% Sort the signal and probability vectors based on ascending order of probability
[s, i] = sort(prob);
huff_tree = huff_tree(i);

huff_tree = create_tree(huff_tree, n_ary, variance); % create a Huffman tree
[huff_tree,dict,avglen] = create_dict(huff_tree,{},0, n_ary); % create the codebook


% Sort the dictionary
[dictsort,dictsortorder] = sort([dict{:,4}]);
lenDict = length(dictsortorder);
finaldict = cell(lenDict, 2);
for i=1:length(dictsortorder)
    finaldict{i,1} = dict{dictsortorder(i), 1};
    finaldict{i,2} = dict{dictsortorder(i), 2};
end
dict = finaldict;



% Recursive function to create the Huffman Code Tree
function huff_tree = create_tree(huff_tree, n_ary, variance)

% If tree length = 1, terminate loop (last node)
if( length(huff_tree) <= 1)
    return;
end

% Combine first 2 nodes under new parent, remove them from list, add new parent
temp = struct('signal', [], 'probability', 0, 'child', [], 'code', []);

for i=1:n_ary
    if isempty(huff_tree), break; end
    temp.probability = temp.probability + huff_tree(1).probability; % Order: ASC
    temp.child{i} = huff_tree(1);
    temp.origOrder = -1;
    huff_tree(1) = [];
end
```

```
if( strcmpi(variance, 'min') == 1 )
    huff_tree = insertMinVar(huff_tree, temp);
else
    huff_tree = insertMaxVar(huff_tree, temp);
end

% Create Huffman tree from the reduced number of free nodes
huff_tree = create_tree(huff_tree, n_ary, variance);
return;



% Recursive function to scan tree to create the codes for each leaf node.
function [huff_tree,dict,total_wted_len] = create_dict(huff_tree,dict,total_wted_len,
n_ary)

% Check if node is leaf->add signal on this node and its corresponding code to dict
if isempty(huff_tree.child)
    dict{end+1,1} = huff_tree.signal;
    dict{end, 2} = huff_tree.code;
    dict{end, 3} = length(huff_tree.code);
    dict{end, 4} = huff_tree.origOrder;
    total_wted_len = total_wted_len + length(huff_tree.code)*huff_tree.probability;
    return;
end

num_childrens = length(huff_tree.child);
for i = 1:num_childrens
    huff_tree.child{i}.code = [huff_tree(end).code, (num_childrens-i)];
    [huff_tree.child{i}, dict, total_wted_len] = create_dict(huff_tree.child{i}, dict,
total_wted_len, n_ary);
end



% Add node in sorted list to convert to ASC. If node with same prob exists
% place new, after it
function huff_tree = insertMaxVar(huff_tree, newNode)

i = 1;
while i <= length(huff_tree) && newNode.probability > huff_tree(i).probability
    i = i+1;
end
huff_tree = [huff_tree(1:i-1) newNode huff_tree(i:end)];



% Add node in sorted list to convert to ASC. If node with same prob exists
% place new, before it
function huff_tree = insertMinVar(huff_tree, newNode)

i = 1;
while i <= length(huff_tree) && newNode.probability >= huff_tree(i).probability
    i = i+1;
end
huff_tree = [huff_tree(1:i-1) newNode huff_tree(i:end)];
```

my_henco_.m

```
function [ enco ] = my_henco(src, dict)
% Erotima 1/1/(b) Code
```

```
% Check if the input is a vector
[m,n] = size(src);
if (m ~= 1 && n ~= 1)
    disp('Input source was not a vector!');
end

% Convert input source to cell array, if it wasn't already
if (~iscell(src) )
    [m,n] = size(src);
    src = mat2cell(src, ones(1,m), ones(1,n) );
end

% Find the size of the largest element in the dict & preallocate 'enco'
maxSize = 0;
dictSize = size(dict,1);
for i = 1:dictSize
    tempSize = size(dict{i,2},2);
    if (tempSize > maxSize)
        maxSize = tempSize;
    end
end
enco = zeros(1, length(src)*maxSize);

% For each given signal value, search through the dictionary to find its code
sigCode = 1;
for i = 1:length(src)
    tempCode = [];
    for j = 1:dictSize
        if (src{i} == dict{j,1})
            tempCode = dict{j,2};
            break;
        end
    end

    codeLen = length(tempCode);
    if (codeLen ~= 0)
        enco(sigCode : sigCode+codeLen-1) = tempCode;
        sigCode = sigCode + codeLen;
    end
end
enco = enco(1:sigCode-1);

% if input was a column vector, transpose the encoded signal vector
if (n == 1)
    enco = enco';
end
```

my_hdeco.m

```
function [ deco ] = my_hdeco(encod, dict)
% Erotima 1/1/(c) Code

% Check if the input is a vector
[m,n] = size(encod);
if ( m ~= 1 && n ~= 1)
    disp('Input was not a vector!');
end



i = 1;
deco = {};
while (i <= length(encod))
    tempCode = encod(i);
    foundCode = find_Code(tempCode, dict);
```

```
    while (isempty(foundCode) && i < length(encod))
        i = i+1;
        tempCode = [tempCode, encod(i)];
        foundCode = find_Code(tempCode, dict);
    end

    if( i == length(encod) && isempty(foundCode) )
        disp('Code not found in the dictionary');
    end

    deco{end+1} = foundCode;
    i=i+1;
end

% if input was a column vector, transpose the encoded signal vector
if( n == 1 )
    deco = deco';
end



% Compare the code with the dictionary entries and return the signal if it exists
function [ exists ] = find_Code(code, dict)

exists = [];
m = size(dict);
for i=1:m(1)
    if ( isequal(code, dict{i,2}) )
        exists = dict{i,1};
        return;
    end
end
```

ask1_2.m

```
% Erotima 1/2 Code

clc; % Clear the Command Window


% Alphabet & Symbol probabilities of Sources A & B
abet1 = cellstr(('a':'z')');
prob1 = [0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015, 0.06094, ...
    0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749, 0.07507, 0.01929, 0.00095,
...
    0.05987, 0.06327, 0.09056, 0.02758, 0.00978, 0.02361, 0.00150, 0.01974, 0.00074];

% Create Source A (10000 random lowercase characters)
src1 = char(randsrc(10000,1,[(97:122); prob1]));

% Begin Huffman decoding of Source A
[dict1, len1] = my_hdict(abet1, prob1);
enco1 = my_henco(src1, dict1);
deco1 = char(my_hdeco(enco1, dict1));

% Print success / failure message onscreen
if (isequal(src1,deco1))
    fprintf(1, 'Huffman decoding of Source A was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len1);
else
    fprintf(1, 'Huffman decoding of Source A was not successful!\n');
end
```

```
% Read Source B from file 'kwords.txt'
% Convert Uppercase letters to Lowercase and remove extra characters
src2 = lower(fscanf(fopen('kwords.txt'),'%s'));
src2(regexp(src2,'[''/.-]'))=[];
src2 = src2.';

% Begin Huffman decoding of Source B
[dict2, len2] = my_hdict(abet1, prob1);
enco2 = my_henco(src2, dict2);
deco2 = char(my_hdeco(enco2, dict2));

% Print success / failure message onscreen
if (isequal(src2,deco2))
    fprintf(1, 'Huffman decoding of Source B was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len2);
else
    fprintf(1, 'Huffman decoding of Source B was not successful!\n');
end
```

ask1_3.m

```
% Erotima 1/3 Code

clc; % Clear the Command Window


% Read Source B from file 'kwords,txt'
srcb = fscanf(fopen('kwords.txt'),'%s');

% Find alphabet of Source B
abet = (unique(srcb)).';


% Compute probability of each alphabet symbol of Source B
i = 1;
prob = zeros(1, length(abet));
for j = 1:length(abet)
    ch = strfind(srcb, abet(j));
    prob(i) = length(ch)/length(srcb);
    i = i+1;
end


% Begin Huffman decoding of Source B
srcb = srcb.';
abet = cellstr(abet);
[dict, len] = my_hdict(abet, prob);
enco = my_henco(srcb, dict);
deco = char(my_hdeco(enco, dict));

% Print success / failure message onscreen
if (isequal(srcb,deco))
    fprintf(1, 'Huffman decoding of Source B was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len);
else
    fprintf(1, 'Huffman decoding of Source B was not successful!\n');
end
```

ask1_4.m

```
% Erotima 1/4 Code
```

```
clc; % Clear the Command Window


% Initial Alphabet & Symbol probabilities of Source A
abet0 = ('a':'z');
prob0 = [0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015, 0.06094, ...
    0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749, 0.07507, 0.01929, 0.00095,
...
    0.05987, 0.06327, 0.09056, 0.02758, 0.00978, 0.02361, 0.00150, 0.01974, 0.00074];

% New Alphabet of Source A
abet = nchoosek(['a':'z', 'a':'z'], 2);
abet = char(unique(cellstr(abet)).');

% Compute probability of each pair (alphabet symbol) of Source A
prob = zeros(length(abet),1);
for i = 1:length(abet)
    ch1 = strfind(abet0, abet(i,1));
    ch2 = strfind(abet0, abet(i,2));
    prob(i,1) = prob0(1,ch1) * prob0(1,ch2);
end

% Create Source A (5000 pairs of random lowercase characters)
srca1 = char(randsrc(5000,1,[(97:122); prob0]));
srca2 = char(randsrc(5000,1,[(97:122); prob0]));
srca = strcat(srca1, srca2);


% Begin Huffman decoding of Source A
abet = cellstr(abet);
srca = cellstr(srca);
set(0,'RecursionLimit',1000);
[dict, len] = my_hdict(abet, prob);
enco = my_henco(srca, dict);
deco = my_hdeco(enco, dict);

% Print success / failure message onscreen
if (isequal(srca,deco))
    fprintf(1, 'Huffman decoding of Source A was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len);
else
    fprintf(1, 'Huffman decoding of Source A was not successful!\n');
end
```

ask1_5.m

```
% Erotima 1/5 Code

clc; % Clear the Command Window


% Read Source B from file 'kwords,txt'
srcb = char(fread(fopen('kwords.txt')));


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


% Initial Alphabet & Symbol probabilities of Source A
abet0a = ('a':'z');
prob0a = [0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015, 0.06094, ...
    0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749, 0.07507, 0.01929, 0.00095,
...
```

```
    0.05987, 0.06327, 0.09056, 0.02758, 0.00978, 0.02361, 0.00150, 0.01974, 0.00074];

% New Alphabet of Source A
abet1 = nchoosek(['a':'z', 'a':'z'], 2);
abet1 = char(unique(cellstr(abet1)).');

% Compute probability of each pair of Source A alphabet
prob1 = zeros(length(abet1),1);
for i = 1:length(abet1)
    ch1 = strfind(abet0a, abet1(i,1));
    ch2 = strfind(abet0a, abet1(i,2));
    prob1(i,1) = prob0a(1,ch1) * prob0a(1,ch2);
end

% Convert Uppercase letters to Lowercase and remove extra characters of Source B
srcb1 = lower(fscanf(fopen('kwords.txt'),'%s'));
srcb1(regexp(srcb1,'[''/.-]'))=[];
srcb1 = cellstr(reshape(srcb1,2,[])');


% Begin Huffman decoding of Source B with probabilities of Erotima 1/4
abet1 = cellstr(abet1);
set(0,'RecursionLimit',1000);
[dict1, len1] = my_hdict(abet1, prob1);
enco1 = my_henco(srcb1, dict1);
deco1 = my_hdeco(enco1, dict1);

% Print success / failure message onscreen
if (isequal(srcb1,deco1))
    fprintf(1, 'Huffman decoding of Source B was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len1);
else
    fprintf(1, 'Huffman decoding of Source B was not successful!\n');
end


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


% Find initial alphabet of Source B
abet0b = unique(srcb).';

% Compute initial probability of each alphabet symbol of Source B
i = 1;
prob0b = zeros(1, length(abet0b));
for j = 1:length(abet0b)
    ch = strfind(srcb.', abet0b(j));
    prob0b(i) = length(ch)/length(srcb);
    i = i+1;
end

% New Alphabet of Source B
abet2 = nchoosek([abet0b, abet0b], 2);
abet2 = char(unique(cellstr(abet2)));

% Split Source B in 2-char cells & find new alphabet
%srcb2 = cellstr(reshape(srcb,2,[])');
%abet2 = char(unique(srcb2));

% Compute new probability of each pair of symbols of Source B
prob2 = zeros(length(abet2),1);
for i = 1:length(abet2)
    ch3 = strfind(abet0b, abet2(i,1));
    ch4 = strfind(abet0b, abet2(i,2));
    prob2(i,1) = prob0b(1,ch3) * prob0b(1,ch4);
```

```
end


% Begin Huffman decoding of Source B with probabilities of Erotima 1/5
abet2 = cellstr(abet2);
srcb2 = cellstr(srcb2);
[dict2, len2] = my_hdict(abet2, prob2);
enco2 = my_henco(srcb2, dict2);
deco2 = my_hdeco(enco2, dict2);

% Print success / failure message onscreen
if (isequal(srcb2,deco2))
    fprintf(1, 'Huffman decoding of Source B was successful!\n');
    fprintf(1, 'Average codeword length of the Huffman code : %f\n', len2);
else
    fprintf(1, 'Huffman decoding of Source B was not successful!\n');
end
```

my_quantizer.m

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

quant_levels = 2^N; % epipeda kvantismou
vima = (max_value - min_value)/quant_levels;    % vima kvantismou
xq = zeros(length(x),1); % arxiko to dianysma me to shma eksodou me vash th diastash
toy dianismatos eisodou
centers = min_value + vima/2; % ypologizw to kentrou tou prwtou epipedou
p = zeros(1,quant_levels);  %arxiko to mhtrwo me tis pithanothtes emfanishs

% ypologizw ta kentra kvantishs
for i=1:1:quant_levels-1
    centers(i+1) = centers(i) + vima;
end

% kvantisw tis times toy shmatos
for i=1:1:length(x)
    for j=1:1:quant_levels
        if x(i)<=min_value + j*vima;
            xq(i) = centers(j);
            p(j) = p(j) + 1;
            break;
        end
    end
end

% ypologizw tis pithanothtes
p = p./length(x);

end
```

Lloyd_Max.m

```
function [xq, centers, D, p] = Lloyd_Max(x,N,min_value,max_value)
kmax = 0;
quant_levels = 2^N; % ypologizw ta epipeda kvantismou me vash to N
xq = zeros(length(x),1) ; % arxikopoiw to kvantismeno shma me mhdenika
centers = zeros(quant_levels,1) ; % arxikopoiw ta kentra kvantismou me mhdenika
D(1) = 0;
Sqnr(1) = 0;
p = zeros(quant_levels,1) ; % arxikopoiw to mhtrwo pou tha ekxwrisw tis pithanothtes
d = (max_value - min_value)/quant_levels ; % ypologizw to vima kvantismou
centers(1) = min_value + d/2 ; % ypologizw to prwto kentro
```

```
% ypologismos twn kentrwn kvantishs
for i =1:quant_levels-1
    centers(i+1) = centers(i) + d ; %ypologismos kentrwn perioxwn
end

% diadikasia kvantismou toy shmatos
while 1
    kmax = kmax + 1; % afksanw ton metrhth epanalipshs
    T(quant_levels+1) = max_value ; % arxikopoiw to panw orio
    T(1) = min_value;   % arxikopoiw to katw orio
    sum =zeros(quant_levels) ;
    counter=zeros(quant_levels) ; %arxikopoihsh counter emfanisewn
    for k=2:quant_levels
        T(k)=(centers(k-1)+centers(k))/2 ; %prosdiorismos orion
        %perioxon kvantisis
    end
    if (x(kmax)>=max_value) %Diadikasi kbantishs
    xq(kmax, 1) = 1;
    elseif (x(kmax)<min_value)
        xq(kmax, 1) = quant_levels ;
    else
        for i =1:length(x)
            for k=1:quant_levels
                if x(i)>T(k) && x(i)<=T(k+1)
                    p(k) = p(k) + 1;
                    xq(i, 1)=centers(k) ;
                    sum(k)=sum(k)+x(i) ;
                    counter(k)=counter(k)+1;
                    break;
                end
            end
        end
    end
    for k=1:quant_levels
        p(k) = p(k)/length(x) ;
        if (counter(k) > 0 )
            centers(k) = sum(k)/counter(k) ; %Ypologismos newn kentrwn
        end
    end
    % ypologizw thn paramorfosh se kathe epanalipsh
    D(kmax+1)=mean((x-xq).^2);
    Sqnr(kmax) = mean(x.^2)/ D(kmax+1);
    % kai an einai mikroterh apo to 10^-16 tote stamataei h diadikasia
    a = 10^-16;
    if (abs(D(kmax+1)-D(kmax))<a)
        break;
    end
end
```

sqnr.m

```
function [sqnr_exp, sqnr_theor] = sqnr(x, xq, N)
%prwta ypologizoume thn peiramatiki timh toy sqnr
sfalma = abs(xq-x);
sqnr_exp = 10*log10(sum(x.^2)/sum(sfalma.^2));

% kai sthn synexeia ypologizoume thn theoritikh timw toy sqnr
sqnr_theor = 1.76+6.02*N;

end
```

erotima2b.m

```
x =
wavread('speech.wav');
                        xq2my = my_quantizer(y,2,-1,1);
                        sqnr2my = sqnr(y,xq2my,2);
                        xq2lloyd = Lloyd_Max(y,2,-1,1);
                        sqnr2lloyd = sqnr(x,xq2lloyd,2);

                        xq4my = my_quantizer(y,4,-1,1);
                        sqnr4my = sqnr(y,xq4my,4);
                        xq4lloyd = Lloyd_Max(y,4,-1,1);
                        sqnr4lloyd = sqnr(x,xq4lloyd,4);

                        xq6my = my_quantizer(y,6,-1,1);
                        sqnr6my = sqnr(y,xq6my,6);
                        xq6lloyd = Lloyd_Max(y,6,-1,1);
                        sqnr6lloyd = sqnr(y,xq6lloyd,6);
```

erotima2c.m

```
y =
wavread('speech.wav');
                        [xq2lloyd,centers] = Lloyd_Max(y,2,-1,1);
                        [xq4lloyd,centers] = Lloyd_Max(y,4,-1,1);
                        [xq6lloyd,centers] = Lloyd_Max(y,6,-1,1);


                        freq2bits_exp = exp_freq(xq2lloyd);
                        freq4bits_exp = exp_freq(xq4lloyd);
                        freq6bits_exp = exp_freq(xq6lloyd);
```

exp_freq.m

```
function
[ freq ]
=
exp_freq(
xq )
            uniq = unique(xq);
            freq = [uniq,histc(xq(:),uniq)];
            freq(:,2) = freq(:,2)./length(xq);
            freq(:,1) = [];
            disp(freq);

            end
```

erotima2d.m

```
mse1 = immse(y, xq2lloyd);
mse2 = immse(y, xq4lloyd);
```

```
mse3 = immse(y, xq6lloyd);
mse4 = immse(y, centers4(xq2my));
mse5 = immse(y, centers5(xq4my));
mse6 = immse(y, centers6(xq6my));
fprintf(1, '\nThe Mean Square Error of Lloyd_Max, Source B, for N=2 is: %f\n', mse1);
fprintf(1, 'The Mean Square Error of my_quantizer, Source B, for N=2 is: %f\n', mse4);
fprintf(1, '\nThe Mean Square Error of Lloyd_Max, Source B, for N=4 is: %f\n', mse2);
fprintf(1, 'The Mean Square Error of my_quantizer, Source B, for N=4 is: %f\n', mse5);
fprintf(1, '\nThe Mean Square Error of Lloyd_Max, Source B, for N=6 is: %f\n', mse3);
fprintf(1, 'The Mean Square Error of my_quantizer, Source B, for N=6 is: %f\n', mse6);
```

mainb.m

```
Tsample=1;
Tc=4;
fc=1/Tc;
Tsymbol=40;
Lb = 100002;
z=0;

for SNR=0:5:40
    %Mapper
    in=mapper(input,M,gray);

    %rectangular pulse
    g=sqrt(2/Tsymbol);

    %M-PAM modulation
    A=1/sqrt(M+1);
    for m=0:M-1
        s(m+1)=(2*m-1-M)*A;
    end

    for i=1:length(in)
        for t=1:Tsymbol
            signal(i,t)=s(in(i)+1)*g*cos(2*pi*fc*t);
        end
    end

    %AWGN
    sigma_sq=(1/log2(M))/(2*10^(SNR/10)); %?^2
    noise=sqrt(sigma_sq)*randn(Lb/log2(M),Tsymbol); %Gaussian noise

    send=signal+noise;

    %M-PAM demodulation
    for t=1:Tsymbol
        y(t)=g*cos(2*pi*fc*t);
    end
    r=send*y';

    %Envelope Detector
    for i=1:length(r)
        for j=1:M
            temp(i,j)=norm(r(i)-s(j));
        end
        s_h(i)=min(temp(i,:));
        for j=1:M
            if s_h(i)==temp(i,j)
                s_hat(i)=j-1;
            end
        end
```

```
    end

    %Demapper
    output=demapper(s_hat,M,gray);

    z=z+1;
    %BER calculation
    b_errors=0;
    for i=1:length(output)
        if output(i)~=input(i)
            b_errors=b_errors +1 ;
        end
    end
    BER(z)=b_errors/length(input);

    %SER calculation
    s_errors=0;
    for i=1:length(s_hat)
        if s_hat(i)~=in(i)
            s_errors=s_errors +1 ;
        end
    end
    SER(z)=s_errors/length(in);
end
```

mapper.m

```
function [y] = mapper(x,M,gray)
k = 1;
if gray == 0 %Simple encoding
    if M == 2
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
            elseif x(i) == 0
                y(k) = 1;
            end
            k = k+1 ;
        end
    elseif M == 4
        for i = 1:2:length(x)
            if x(i) == 0 && x(i+1) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 1
                y(k) = 1;
            elseif x(i) == 1 && x(i+1) == 0
                y(k) = 2;
            elseif x(i) == 1 && x(i+1) == 1
                y(k) = 3;
            end
            k = k+1 ;
        end
    elseif M == 8
        for i = 1:3:length(x)
            if x(i) == 0 && x(i+1) == 0 && x(i+2) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1
                y(k) = 1;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0
                y(k) = 2;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1
                y(k) = 3;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0
                y(k) = 4;
```

```
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1
                y(k) = 5;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0
                y(k) = 6;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1
                y(k) = 7;
            end
            k = k+1;
        end
       elseif M == 16
        for i = 1:4:length(x)
            if x(i) == 0 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 1;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 2;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 3;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 4;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 5;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 6;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 7;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 8;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 9;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 10;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 11;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 12;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 13;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 14;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 15;
            end
            k = k+1;
        end
    end
elseif gray == 1 %Gray code
    if M == 2
        for i = 1:1:length(x)
            if x(i) == 0 && x(i+1) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 1
                y(k) = 1;
            end
            k = k+1 ;
        end
    elseif M == 4
        for i = 1:2:length(x)
            if x(i) == 0 && x(i+1) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 1
                y(k) = 1;
            elseif x(i) == 1 && x(i+1) == 1
                y(k) = 2;
            elseif x(i) == 1 && x(i+1) == 0
                y(k) = 3;
```

```
            end
            k = k+1 ;
        end
    elseif M == 8
        for i = 1:3:length(x)
            if x(i) == 0 && x(i+1) == 0 && x(i+2) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1
                y(k) = 1;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1
                y(k) = 2;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0
                y(k) = 3;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0
                y(k) = 4;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1
                y(k) = 5;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1
                y(k) = 6;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0
                y(k) = 7;
            end
            k = k+1;
        end
         elseif M == 16
        for i = 1:4:length(x)
            if x(i) == 0 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 0;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 1;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 2;
            elseif x(i) == 0 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 3;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 4;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 5;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 6;
            elseif x(i) == 0 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 7;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 8;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 9;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 10;
            elseif x(i) == 1 && x(i+1) == 1 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 11;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 0
                y(k) = 12;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 1 && x(i+3) == 1
                y(k) = 13;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 1
                y(k) = 14;
            elseif x(i) == 1 && x(i+1) == 0 && x(i+2) == 0 && x(i+3) == 0
                y(k) = 15;
            end
            k = k+1;
        end
     end
end
```

demapper.m

```
function [y] = demapper(x,M,gray)
k = 1;
if gray == 0 %Simple encoding
    if M == 2
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
            elseif x(i) == 1
                y(k) = 1;
            end
            k = k+1 ;
        end
    elseif M == 4
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
                y(k+1) = 0;
            elseif x(i) == 1
                y(k) = 0;
                y(k+1) = 1;
            elseif x(i) == 2
                 y(k) = 1;
                y(k+1) = 0;
            elseif x(i) == 3
                y(k) = 1;
                y(k+1) = 1;
            end
            k = k+2 ;
        end
    elseif M == 8
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
                y(k+1) = 0;
                y(k+2)=0;
            elseif x(i) == 1
                y(k) = 0;
                y(k+1) = 0;
                y(k+2)=1;
            elseif x(i) == 2
                 y(k) = 0;
                y(k+1) = 1;
                y(k+2)=0;
            elseif x(i) == 3
                y(k) = 0;
                y(k+1) = 1;
                y(k+2)=1;
            elseif x(i) == 4
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=0;
            elseif x(i) == 5
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=1;
            elseif x(i) == 6
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=0;
            elseif x(i) == 7
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=1;
            end
            k = k+3 ;
        end
        elseif M == 16
```

```
for i = 1:length(x)
    if x(i) == 0
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=0;
         y(k+3)=0;
    elseif x(i) == 1
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=0;
        y(k+3)=1;
    elseif x(i) == 2
         y(k) = 0;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=0;
    elseif x(i) == 3
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=1;
    elseif x(i) == 4
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=0;
    elseif x(i) == 5
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=1;
    elseif x(i) == 6
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=1;
        y(k+3)=0;
    elseif x(i) == 7
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=1;
        y(k+3)=1;
    elseif x(i) == 8
        y(k) = 1;
        y(k+1) = 0;
        y(k+2)=0;
        y(k+3)=0;
    elseif x(i) == 9
         y(k) = 1;
        y(k+1) = 0;
        y(k+2)=0;
        y(k+3)=1;
    elseif x(i) == 10
        y(k) = 1;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=0;
    elseif x(i) == 11
        y(k) = 1;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=1;
    elseif x(i) == 12
        y(k) = 1;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=0;
    elseif x(i) == 13
```

```
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=0;
                y(k+3)=1;
            elseif x(i) == 14
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=1;
                y(k+3)=0;
            elseif x(i) == 15
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=1;
                y(k+3)=1;
            end
            k = k+4 ;
        end
    end
elseif gray == 1 %Gray code
    if M == 2
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
            elseif x(i) == 1
                y(k) = 1;
            end
            k = k+1 ;
        end
    elseif M == 4
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
                y(k+1) = 0;
            elseif x(i) == 1
                y(k) = 0;
                y(k+1) = 1;
            elseif x(i) == 2
                 y(k) = 1;
                y(k+1) = 1;
            elseif x(i) == 3
                y(k) = 1;
                y(k+1) = 0;
            end
            k = k+2 ;
        end
    elseif M == 8
        for i = 1:length(x)
            if x(i) == 0
                y(k) = 0;
                y(k+1) = 0;
                y(k+2)=0;
            elseif x(i) == 1
                y(k) = 0;
                y(k+1) = 0;
                y(k+2)=1;
            elseif x(i) == 2
                 y(k) = 0;
                y(k+1) = 1;
                y(k+2)=1;
            elseif x(i) == 3
                y(k) = 0;
                y(k+1) = 1;
                y(k+2)=0;
            elseif x(i) == 4
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=0;
```

```
    elseif x(i) == 5
        y(k) = 1;
        y(k+1) = 1;
        y(k+2)=1;
    elseif x(i) == 6
        y(k) = 1;
        y(k+1) = 0;
        y(k+2)=1;
    elseif x(i) == 7
        y(k) = 1;
        y(k+1) = 0;
        y(k+2)=0;
    end
    k = k+3 ;
end
 elseif M == 16
for i = 1:length(x)
    if x(i) == 0
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=0;
         y(k+3)=0;
    elseif x(i) == 1
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=0;
        y(k+3)=1;
    elseif x(i) == 2
         y(k) = 0;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=1;
    elseif x(i) == 3
        y(k) = 0;
        y(k+1) = 0;
        y(k+2)=1;
        y(k+3)=0;
    elseif x(i) == 4
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=1;
        y(k+3)=0;
    elseif x(i) == 5
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=1;
        y(k+3)=1;
    elseif x(i) == 6
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=1;
    elseif x(i) == 7
        y(k) = 0;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=0;
    elseif x(i) == 8
        y(k) = 1;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=0;
    elseif x(i) == 9
         y(k) = 1;
        y(k+1) = 1;
        y(k+2)=0;
        y(k+3)=1;
```

```
            elseif x(i) == 10
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=1;
                y(k+3)=1;
            elseif x(i) == 11
                y(k) = 1;
                y(k+1) = 1;
                y(k+2)=1;
                y(k+3)=0;
            elseif x(i) == 12
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=1;
                y(k+3)=0;
            elseif x(i) == 13
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=1;
                y(k+3)=1;
            elseif x(i) == 14
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=0;
                y(k+3)=1;
            elseif x(i) == 15
                y(k) = 1;
                y(k+1) = 0;
                y(k+2)=0;
                y(k+3)=0;
            end
            k = k+4 ;
        end
    end
end
```

myresult.m

```
Lb = 100002;
input = randsrc(Lb,1,[0 1]);

figure (1) %BER
gray=0;
M=2;
mainb;
SNR=0:5:40;
semilogy(SNR,BER,'bo-'); xlabel('SNR'); ylabel('BER');
clearvars -except input;


gray=0;
M=4;
mainb;
SNR=0:5:40;
hold on
semilogy(SNR,BER,'b^-'); xlabel('SNR'); ylabel('BER');
hold off
clearvars -except input;

gray=1;
M=4;
mainb;
SNR=0:5:40;
hold on
```

```
semilogy(SNR,BER,'ro-'); xlabel('SNR'); ylabel('BER');
hold off
clearvars -except input;

gray=0;
M=8;
mainb;
SNR=0:5:40;
hold on
semilogy(SNR,BER,'r^-'); xlabel('SNR'); ylabel('BER');
hold off
clearvars -except input;


gray=1;
M=8;
mainb;
SNR=0:5:40;
hold on
semilogy(SNR,BER,'co-'); xlabel('SNR'); ylabel('BER');
hold off
clearvars -except input;

gray=0;
M=16;
mainb;
SNR=0:5:40;
hold on
semilogy(SNR,BER,'c^-'); xlabel('SNR'); ylabel('BER');
hold off
clearvars -except input;

gray=1;
M=16;
mainb;
SNR=0:5:40;
hold on
semilogy(SNR,BER,'z^-'); xlabel('SNR'); ylabel('BER');
hold off
legend('2PAM','4PAM','4PAM-Gray','8PAM','8PAM-Gray','16PAM','16PAM-Gray');
clearvars -except input;
```