Τμημα Μηχανικων Ηλεκτρονικων Υπολογιστων και Πληροφορικης

Πολυτεχνικη Σχολη

Ψηφιακές Τηλεπικοινωνίες - Πρώτη Εργαστηριακή Άσκηση 2021-2022

Μία εργασία

του

Νικόλαου Σκαμνέλου

Α.Μ: 1041878

Έτος: 8ο

Πάτρα 19/1/2022

Contents

1 Ερώτημα 1 - Κωδικοποίηση Huffman **2**

1.1 Κωδικοποίηση Huffman . . . 2

1.1.1 Ζητούμενο 1 . . . 3

1.1.2 Ζητούμενο 2 . . . 4

1.1.3 Ζητούμενο 3 . . . 6

1.1.4 Ζητούμενο 4 . . . 6

1.1.5 Ζητούμενο 5 . . . 7

2 Ερώτημα 2 - Κωδικοποίηση PCM **9**

2.1 Κωδικοποίηση PCM . . . 9

2.1.1 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής . . . 9

2.1.2 Μη Ομοιόμορφος Διανυσματικός Κβαντιστής . . . 11

2.1.3 Υπολογισμός SQNR - MSE . . . 12

2.1.4 Ζητούμενο 1 . . . 13

2.1.5 Ζητούμενο 2 . . . 14

3 Κώδικες Υλοποίησης MATLAB **17**

Chapter 1

Ερώτημα 1 - Κωδικοποίηση Huffman

1.1 Κωδικοποίηση Huffman

Αρχικά, θα γίνει αναφορά στα κύρια χαρακτηριστικά της κωδικοποίησης Huffman. Η κωδικοποίησή Huffman είναι μία μορφή προθεματικού κώδικα, η οποία χρησιμοποιείται κυρίως για συμπίεση δεδομένων, χωρίς απώλεια πληροφορίας (lossless compression). Αναπτύχθηκε από τον David Huffman το 1952 και η διαδικασία παραγωγής του είναι η εξής:

- Αρχικά, τα σύμβολα του αλφάβητου ταξινομούνται φθίνουσα σε μία λίστα, σύμφωνα με την πιθανότητα εμφάνισης τους.
- Εντοπίζονται τα δύο σύμβολα με τις χαμηλότερες πιθανότητες εμφάνισης και συγχωνεύονται. Το παραγόμενο σύμβολο έχει πιθανότητα εμφάνισης ίση με το άθροισμα των επιμέρους συμβόλων.
- Στα συνιστώσα σύμβολα αναθέτονται οι τιμές "0" και "1".
- Γίνεται ξανά φθίνουσα ταξινόμησης της λίστας με τα σύμβολα σύμφωνα με την ελάχιστη πιθανότητα εμφάνισης.
- Γίνεται επανάληψη της παραπάνω διαδικασίας μέχρι να υπάρχει μόνο ένα σύμβολο στην λίστα.

Για μία πηγή με N σύμβολα, η διαδικασία που αναφέρθηκε παραπάνω, θα επαναληφθεί N-1 φορές. Επίσης, παρατηρείτε ότι εκτελώντας την παραπάνω διαδικασία παράγεται ένα δυαδικό δέντρο, του οποίου η ρίζα είναι το τελικό ενοποιημένο σύμβολο, τα φύλλα είναι τα αρχικά σύμβολα και οι ενδιάμεση κόμβοι είναι τα υπόλοιπα ενοποιημένα σύμβολα. Η κωδικοποίηση Huffman για κάποιο σύμβολο είναι η ακολουθία από 0 και 1 που παράγεται αν ξεκινώντας από την ρίζα του δυαδικού δέντρου κινηθούμε ως προς το φύλλο που αντιστοιχεί στο επιθυμητό σύμβολο.

Η κωδικοποίηση Huffman, αν και απαιτεί εκ τον προτέρων γνώση των πιθανοτήτων εμφάνισης των συμβόλων, πετυχαίνει το ελάχιστο μέσο μήκος κωδικοποίησης από όλους τους προθεματικούς symbol-by-symbol κώδικες.

1.1.1 Ζητούμενο 1

Αρχικά για την κωδικοποίηση Huffman πρέπει να δημιουργηθεί ένα λεξικό το οποίο να περιέχει τα σύμβολα που κωδικοποιούνται και του κωδικούς που αναθέτονται σε κάθε από αυτά τα σύμβολα. Αυτό επιτυγχάνεται ακολουθώντας την διαδικασία που περιγράφηκε στην αρχή του κεφαλαίου. Χρησιμοποιήθηκε μία συνάρτηση με όνομα huffmandict, η οποία δέχεται ως είσοδο ένα πίνακα που περιέχει τα σύμβολα για τα οποία θα παραχθεί η κωδικοποίηση huffman, καθώς και έναν πίνακα με τις αντίστοιχες πιθανότητες για κάθε σύμβολο. Επιστρέφεται το λεξικό (dict), που αναφέρθηκε στην αρχή της παραγράφου και το μέσο μήκος (avglen) του παραγόμενου κώδικα huffman. Είναι αναγκαίο να σημειωθεί ότι η συγκεκριμένη συνάρτηση έχει την ικανότητα να παράξει λεξικά και για επεκταμένες πηγές. Η υλοποίηση της συνάρτησης φαίνεται στο 3.1.

Ακολουθεί μία σύντομη περιγραφή του κώδικα. Στις γραμμές 7-20 υπολογίζουμε τις ενώσεις των συμβόλων με τις μικρότερες πιθανότητες και τις αποθηκεύουμε σε ένα πίνακα, ο οποίος λειτουργεί ως χάρτης για τα επόμενα βήματα. Πρέπει να σημειωθεί ότι σε κάθε επανάληψη της διαδικασίας το συγχωνευμένο σύμβολο αντιστοιχεί στον δείκτη 1. Στην συνέχεια, δημιουργείται και αρχικοποιείται το δυαδικό δέντρο (γραμμές 23-28). Υπολογίζονται οι κώδικες huffman για κάθε επίπεδο του δέντρου με την βοήθεια του πίνακα χάρτη (γραμμές 31-42) και εξάγονται οι κωδικοί για κάθε σύμβολο (γραμμές 44-48). Τέλος, δημιουργείται το λεξικό και εισάγονται σε αυτό τα σύμβολα με τους αντίστοιχους κώδικες τους (γραμμές 53-63). Πριν τερματίσει ο αλγόριθμος υπολογίζεται και το μέσο μέγεθος του κώδικα (γραμμές 65-71). Ο τύπος που ακολουθήθηκε για τον υπολογισμό του μέσου μεγέθους κώδικα ήταν αυτό που φαίνεται στο 1.1, όπου $\bar{L}$ είναι το μέσο μέγεθος του κώδικα, $p(s_i)$ είναι η πιθανότητα του σύμβολου s_i και το $l(s_i)$ είναι το μήκος του κωδικού huffman για το σύμβολο s_i.

$$\bar{L} = \sum_{i=1}^{N} p\left(s_i\right) l\left(s_i\right) \tag{1.1}$$

Με την ολοκλήρωση της δημιουργίας του λεξικού, είναι δυνατή η κωδικοποίηση σημάτων. Το μόνο που χρειάζεται να γίνει είναι η αντιστοίχηση των συμβόλων του σήματος με τους κωδικούς huffman. Χρησιμοποιήθηκε μία συνάρτηση με όνομα huffmanenco, που δέχεται ως είσοδο την ακολουθία συμβόλων και το λεξικό με τα σύμβολα και τους κώδικες huffman και επιστέφει την κωδικοποιημένη ακολουθία. Η υλοποίηση αυτής της συνάρτησης φαίνεται στο 3.2. Στην συγκεκριμένη συνάρτηση έχει συμπεριληφθεί και ειδικός κώδικας MATLAB για την κωδικοποίηση δεύτερης επέκτασης πηγής. Αυτό γίνεται αντιστοιχίζοντας δυάδες συμβόλων της ακολουθίας με το λεξικό. Η λειτουργία της συνάρτησης είναι εξαιρετικά απλή. Πρώτα, εξάγονται τα σύμβολα από το λεξικό (γραμμές 9-10), και έπειτα αν έχουμε δεύτερη τάξη επέκτασης πηγής τα σύμβολα κωδικοποιούνται σε ζεύγη (γραμμές 12-27), ειδάλλως κωδικοποιούνται ένα-ένα (γραμμές 28-39).

Η διαδικασία της αποκωδικοποίησης είναι παρόμοια με αυτή της κωδικοποίησης με την διαφορά ότι πλέον γίνεται αντιστοίχηση κωδικών huffman με τα σύμβολα. Όπως και στις παραπάνω περιπτώσεις χρησιμοποιήθηκε μία συνάρτηση με όνομα huffmandeco, που δέχεται ως είσοδο μία ακολουθία από κώδικες huffman και το λεξικό με τα σύμβολα και τους κώδικες και επιστρέφει την ακολουθία σε σύμβολα. Η υλοποίηση αυτής της συνάρτησης φαίνεται στο 3.3. Η διαδικασία που ακολουθήθηκε είναι αυτή που περιγράφηκε στην αρχή της παραγράφου. Πρώτα, εξάγονται η κώδικες huffman από το λεξικό (γραμμές 9-13) και έπειτα γίνεται η αντιστοίχηση των κωδικών huffman σε σύμβολα (γραμμές 15-27, 40-41). Στην περίπτωση της δεύτερης επέκτασης πηγής είναι απαραίτητη η μορφοποίηση της παραγόμενης ακολουθίας των συμβολών (γραμμές 30-38), πράγμα που διασφαλίζει την σωστή μετατροπή της ακολουθίας συμβόλων σε κείμενο.

1.1.2 Ζητούμενο 2

Πριν περάσουμε στα ζητούμενα του ερωτήματος 2, και όλων το επόμενων ερωτημάτων για το κεφάλαιο αυτό, πρέπει πρώτα να αναλυθεί μία συνάρτηση που υλοποιήθηκε για τον υπολογισμό διάφορων μετρικών των παραγόμενων κωδικοποιήσεων. Αυτή η συνάρτηση ονομάζεται huffmanstats και δέχεται ως είσοδο το πίνακα με τις πιθανότητες κάθε συμβόλου και το μέσο μέγεθος κώδικα που υπολογίζεται από την huffmandict και επιστρέφει ένα πίνακα που περιέχει την εντροπία πηγής, το μέσο μέγεθος κώδικα (για σύγκριση) και την αποδοτικότητα της κωδικοποίησης. Η υλοποίηση της συνάρτησης φαίνεται στο 3.4. Η εντροπία της πηγής υπολογίσθηκε σύμφωνα με τον τύπο 1.2, όπου $H(\Phi)$ είναι η εντροπία μίας πηγής Φ, p_i είναι η πιθανότητα εμφάνισης του συμβόλου i και το $I\left(s_i\right)$ είναι η πληροφορία που προσφέρει ένα σύμβολο i.

$$H(\Phi)=\sum_{i=1}^{N} p_i I\left(s_i\right)=-\sum_{i=1}^{N} p_i \log_2 p_i \tag{1.2}$$

Η αποδοτικότητα η της κωδικοποίησης υπολογίσθηκε σύμφωνα με το τύπο 1.3, όπου $H(X)$ είναι η εντροπία της πηγής και $\bar{L}$ είναι το μέσο μέγεθος κώδικα, και και δείχνει πόσο κοντά βρίσκεται ο κωδικοποιητής στο όριο συμπίεσης της πηγής (εντροπία). Όσο πιο κοντά βρίσκεται στο 1 τόσο πιο αποδοτική είναι η κωδικοποίηση.

$$\eta = \frac{H(X)}{\bar{L}} \leq 1 \tag{1.3}$$

Προχωρώντας στο ζητούμενο του ερωτήματος 2, ο κώδικας για την υλοποίηση φαίνεται στο 3.5. Αρχικά, πρέπει να εκτιμηθούν οι πιθανότητες εμφάνισης κάθε συμβόλου (a-z και του κενού). Οι συχνότητες εμφάνισης των γραμμάτων μετριούνται με την βοήθεια ενός ιστογράμματος. Για την συχνότητα εμφάνισης του κενού αρκεί να αφαιρέσουμε από το συνολικό μέγεθος του κειμένου το άθροισμα των εμφανίσεων των γραμμάτων. Έπειτα, υπολογίζονται οι πιθανότητες εμφάνισης κάθε συμβόλου σύμφωνα με τον τύπο 1.4, όπου p_i είναι η πιθανότητα εμφάνισης του συμβόλου i, n_i είναι ο αριθμός εμφανίσεων του συμβόλου i και N είναι το μέγεθος τους κειμένου.

$$p_i = \frac{n_i}{N} \tag{1.4}$$

Αυτά καλύπτουν τις γραμμές 1-21 της υλοποίησης, στην συνέχεια, δημιουργείται ο πίνακα2 με τα σύμβολα του κειμένου και με την βοήθεια της συνάρτησης huffmandict παράγεται το λεξικό με τους κώδικες huffman για κάθε σύμβολο. Επίσης, υπολογίζεται και το μέσο μέγεθος της κωδικοποίησης. Ακολουθεί η κωδικοποίηση του κειμένου, χρησιμοποιώντας την συνάρτηση huffmanenco και μετά η αποκωδικοποίηση του κειμένου με την βοήθεια της συνάρτησης huffmandeco. Έπειτα, γίνεται έλεγχος για την σωστή αποκωδικοποίηση του κειμένου. Αυτά καλύπτονται στις γραμμές 23-38. Τέλος, χρησιμοποιείται η συνάρτηση huffmanstats για τον υπολογισμό μερικών μετρικών. Τα αποτελέσματα της φαίνονται παρακάτω:

```
stats =

  1×3 table

    Entropy    Average Length    efficiency
    _______    ______________    __________

    4.1347         4.163           0.9932
```

Figure 1.1: Source A encoding stats

Η πιο σημαντική παρατήρηση είναι ότι η κωδικοποίηση πετυχαίνει πολύ υψηλά επίπεδα αποδοτικότητας, που πλησιάζουν την μονάδα.

1.1.3 Ζητούμενο 3

Στο συγκεκριμένο ερώτημα, αντί να πρέπει να εκτιμηθούν οι πιθανότητες εμφάνισης, δίνονται σε ένα αρχείο. Ο κώδικας για αυτό το ζητούμενο φαίνεται στο 3.6. Αρχικά, φορτώνουμε το αρχείο με το κείμενο(γραμμές 1-2) και το αρχείο με της πιθανότητες των γραμμάτων της αγγλικής γλώσσας και του κενού (γραμμές 4-5). Εξάγονται οι τιμές των πιθανοτήτων από το αρχείο "frequencies.txt" και μετατρέπονται από αλφαριθμητικά σε αριθμούς (γραμμές 6-13). Στην συνέχεια, δημιουργείται ο πίνακας με τα σύμβολα (γραμμές 15-17) και ακολουθείται η διαδικασία κωδικοποίησης όπως και στο ζητούμενο 1 (γραμμές 19-30). Τέλος, με την βοήθεια της συνάρτησης huffmanstats, υπολογίζονται μερικές μετρικές για την κωδικοποίηση. Τα αποτελέσματα της φαίνονται παρακάτω:

```
stats =

  1×3 table
```

Entropy	Average Length	efficiency
4.1679	4.1875	0.99533

Figure 1.2: Source A encoding stats using english symbol frequencies

Παρατηρείτε μία μικρή βελτίωση στην αποδοτικότητα. Αυτό συμβαίνει μάλλον, επειδή κατά την εκτίμηση των πιθανοτήτων από το κείμενο, το δείγμα είναι πολύ μικρό και δεν έχουμε την ικανότητα να προσεγγίσουμε τις ακριβές πραγματικές πιθανότητες που έχουν τα σύμβολα της αγγλικής γλώσσας. Στο συγκεκριμένο ερώτημα μας δίνονται στο αρχείο οι πραγματικές πιθανότητες οπότε τα αποτελέσματα βελτιώνονται.

1.1.4 Ζητούμενο 4

Στο συγκεκριμένο ζητούμενο θεωρείτε η επέκταση δεύτερης τάξης της πηγής. Οι πιθανότητες κάθε συμβόλου εκτιμήθηκαν με τον ίδιο τρόπο, όπως και στο ζητούμενο 2. Ο κώδικας για το συγκεκριμένο ζητούμενο φαίνεται στο 3.7. Στην αρχή, εκτιμούνται οι πιθανότητές κάθε συμβόλου από το κείμενο όπως και στο 2ο ζητούμενο και έπειτα δημιουργείται ένας πίνακας που περιέχει τα 26 γράμματα του αγγλικού αλφάβητου και το κενό. Με την βοήθεια της εντολής της MATLAB "combvec" παράγονται δύο πίνακες, από τους οποίους, ο πρώτος περιέχει όλους τους πιθανούς συνδυασμούς δύο συμβόλων και ο δεύτερος όλους του πιθανούς συνδυασμούς δύο πιθανοτήτων (γραμμές

27-28). Στην συνέχεια, υπολογίζονται οι τελικές πιθανότητες σύμφωνα με τον τύπο 1.5, όπου σ_K είναι το ζεύγος δύο συμβόλων s_i και s_j (γραμμές 32-34).

$$p\left(\sigma_K\right) = p\left(s_i\right) p\left(s_j\right) \tag{1.5}$$

Το επόμενο βήμα είναι η κωδικοποίηση του κειμένου με την βοήθεια των συναρτήσεων huffmandict, huffmanenco και huffmandeco (γραμμές 36-47). Όπως έχει ήδη αναφερθεί οι συναρτήσεις έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να δουλεύουν και με την δεύτερη τάξη επέκτασης της πηγής. Τέλος, με την βοήθεια της συνάρτησης huffmanstats, υπολογίζονται μερικές μετρικές για την κωδικοποίηση. Τα αποτελέσματα της φαίνονται παρακάτω:

```
stats =

  1×3 table

    Entropy    Average Length    efficiency
    _______    ______________    __________

    8.2694         8.2981          0.99654
```

Figure 1.3: Source A encoding stats using the second extension

Παρατηρείται βελτίωση της αποδοτικότητας σε σχέση και με τις δύο υλοποιήσεις των προηγούμενων ζητούμενων. Επίσης, παρατηρείται διπλασιάσει περίπου και της εντροπίας αλλά και του μέσου μεγέθους του κώδικα. Αυτά ωστόσο είναι αναμενόμενα σύμφωνα με την 1.6, όπου X^n είναι η n-οστή επέκταση της πηγής X.

$$H\left(X^n\right) = nH(X) \tag{1.6}$$

Τέλος, είναι γνωστό ότι η n-οστής τάξης επέκταση μιας πηγής αποφέρει κώδικες που είναι ολοένα και πιο κοντά στο όριο συμπίεσης (εντροπία) της πηγής.

1.1.5 Ζητούμενο 5

Ο κώδικας για το ζητούμενο 5 φαίνεται στο 3.8. Αρχικά, φορτώνουμε την εικόνα cameraman.mat και την διαμορφώνουμε κατάλληλα ώστε να έρθει σε μορφή διανύσματος (γραμμές 1-7). Στην συνέχεια εκτιμούνται οι πιθανότητες εμφάνισις κάθε συμβόλου - αριθμού (γραμμές 9-18) και κωδικοποιείται η πηγή σύμφωνα με την κωδικοποίηση Huffman (γραμμές 20-26). Έπειτα γίνονται μερικές μετατροπές ώστε η κωδικοποίηση να έχει την μορφή διανύσματος (γραμμές 28-41) και μεταδίδεται μέσο του δυαδικού συμμετρικού καναλιού (γραμμές 43-44). Εκτιμάται η πιθανότητα της λάθος και σωστής μετάδοσης (γραμμές 46- 66), και τέλος υπολογίζονται η εντροπία και η χωρητικότητά του καναλιού (γραμμές 68-71).

Chapter 2

Ερώτημα 2 - Κωδικοποίηση PCM

2.1 Κωδικοποίηση PCM

Σκοπός αυτού του κεφαλαίου είναι η υλοποίηση δύο κβαντιστών, έναν μη ομοιόμορφο βαθμωτό και ένα μη ομοιόμορφο διανυσματικό. Στην συνέχεια, θα γίνουν πειράματα με τους συγκεκριμένους κβαντιστές και θα μετρηθεί η απόδοση τους. Πριν φτάσουμε όμως στα πειράματα θα περιγραφεί η υλοποίηση των δύο ζητούμενων κβαντιστών.

2.1.1 Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

Για τον μη ομοιόμορφο βαθμωτό κβαντιστή χρησιμοποιήθηκε ο αλγόριθμος Lloyd - Max, όπως περιγράφεται στην εκφώνηση της άσκησης. Η υλοποίηση του αλγόριθμου φαίνεται στο 3.9. Αρχικά, ο αλγόριθμος υπολογίζει διάφορες μεταβλητές, όπως για παράδειγμα τον αριθμό των επιπέδων (γραμμή 17), το οποίο ισούται με 2^N. Επίσης, φιλτράρει τις τιμές του εισαγομένου σήματος ώστε να μην ξεπερνάνε τα όρια που έχουν ορισθεί (γραμμές 20-21). Επόμενο βήμα, είναι ο προσδιορισμός των αρχικών επιπέδων κβάντισης. Επειδή είναι απαραίτητο τα άκρα των επιπέδων κβάντισης να ισούται με την ελάχιστη και την μέγιστη τιμή του εισαγόμενου σήματος, επιλέγονται τυχαία μόνο M-2 επίπεδα κβάντισης, όπου M είναι ο συνολικός αριθμός των επιπέδων κβάντισης. Τέλος, επειδή οι τιμές που παράγονται είναι τυχαίες ανάμεσα στο διάστημα της ελάχιστης και της μέγιστης τιμής είναι απαραίτητη οι ταξινόμηση των επιπέδων κβάντισης. Τα παραπάνω γίνονται στις γραμμές 27-31.

Έχοντας υπολογίσει όλες τις απαραίτητες μεταβλητές, ο αλγόριθμος είναι έτοιμος για την κύρια διαδικασία υπολογισμού του κβαντισμένου σήματος. Όσο η μέση παραμόρφωση βρίσκεται πάνω από κάποιο όριο epsilon, ο αλγόριθμος αρχικά υπολογίζει τα όρια των ζωνών κβάντισης ως την μέση τιμή δύο διαδοχικών τιμών επιπέδων κβάντισης (γραμμές 43-51). Δηλαδή ακολουθείται ο τύπος 2.1.

$$\mathrm{T}_k = \left(\tilde{x}_k^{(i)} + \tilde{x}_{k+1}^{(i)}\right)/2, \quad 1 \leq k \leq M-1 \tag{2.1}$$

Με την παραπάνω διαδικασία υπολογίζονται M-1 τιμές ζωνών κβάντισης. Έπειτα, αυτές συμπληρώνονται με την ελάχιστη τιμή και μέγιστη τιμή του σήματος καταλήγουμε σε M+1 τιμές ζωνών κβάντισης. Με την ολοκλήρωση του υπολογισμού των ζωνών κβάτνισης, ο αλγόριθμος περνάει στο στάδιο που εντοπίζει σε ποια στάθμη κβάντισης ανήκει κάθε τιμή του εισαγόμενου σήματος. Η διαδικασία είναι εξαιρετικά απλή. Διατρέχεται το εισαγόμενο σήμα, και για κάθε τιμή του, εντοπίζεται το διάστημα κβάντισης στο οποίο ανήκει και αποθηκεύεται σε έναν πίνακα με όνομα xq, ο ακέραιος που αντιστοιχεί στο συγκεκριμένο επίπεδο κβάντισης (γραμμές 54-63). Ιδιαίτερη προσοχή θέλει ο τρόπος που αριθμούνται τα επίπεδα κβάντισης, με το 1 να αντιστοιχεί στο πιο θετικό επίπεδο και το 2^N στο πιο αρνιτικό επίπεδο (γραμμή 59).

Για τον υπολογισμό της μέσης παραμόρφωσης, υπολογίζεται πρώτα η συνολική παραμόρφωση, η οποία ισούται με το άθροισμα των επιμέρους παραμορφώσεων που παρατηρούνται. Επειδή η πηγές που μας δίνονται είναι συνεχούς αλφάβητου, οι παραμορφώσεις υπολογίζονται ως το τετραγωνικό σφάλμα, δηλαδή σύμφωνα με τον τύπο 2.2, όπου $d(x, \hat{x})$ είναι η παραμόρφωση μεταξύ της πραγματικής τιμής x και της κβαντισμένης τιμής $\hat{x}$.

$$d(x, \hat{x}) = (x - \hat{x})^2 \tag{2.2}$$

Η μέση παραμόρφωση υπολογίζεται σύμφωνα με τον τύπο 2.3, όπου n είναι το μέγεθος του σήματος.

$$d\left(\mathbf{x}^n, \hat{\mathbf{x}}^n\right) = \frac{1}{n} \sum_{i=1}^{n} d\left(x_i, \hat{x}_i\right) \tag{2.3}$$

Ο υπολογισμός της μέσης παραμόρφωσης φαίνεται στις γραμμές 65-73 και 76-80 του αλγόριθμου.

Το μόνο που απομένει να γίνει είναι ο υπολογισμός των νέων επιπέδων κβάντισης. Αυτά ισούται με την μέση τιμή των τιμών που ανήκουν σε ένα συγκεκριμένο διάστημα κβάντισης. Δηλαδή ακολουθείτε ο τύπος 2.4, όπου $\widetilde{x}_k^{(i+1)}$ είναι η τιμή του επιπέδου κβάντισης k στην επανάληψη i+1 και T_{k-1}, T_k είναι τα όρια των ζωνών κβάντισης. Ο υπολογισμός αυτών γίνεται στις γραμμές 68-78 και 85-89 του αλγόριθμου.

$$\widetilde{x}_k^{(i+1)} = \mathrm{E}\left[x \mid \mathrm{T}_{k-1} < x < \mathrm{T}_k\right] \tag{2.4}$$

Τέλος, για το ζητούμενο 2 είναι απαραίτητος ο υπολογισμός του SQNR για κάθε επανάληψη του αλγόριθμου (γραμμές 94-95). Η λεπτομερή ανάλυση της υλοποίησης της συνάρτησης με την οποία υπολογίζουμε το SQNR θα γίνει στο υποκεφάλαιο 2.1.3.

Ολοκληρώνοντας τα παραπάνω ο αλγόριθμος είναι έτοιμος να περάσει στην επόμενη επανάληψη. Υπενθυμίζεται ότι η διαδικασία επαναλαμβάνεται ως που να μην υπάρχει βελτίωση στην μέση παραμόρφωση.

2.1.2 Μη Ομοιόμορφος Διανυσματικός Κβαντιστής

Η υλοποίηση του μη ομοιόμορφου διανυσματικού κβαντιστή είναι αρκετά πιο απλή. Χρησιμοποιείται ο αλγόριθμος συσταδοποίησης K-means, ο οποίος εντοπίζει και κατατάσσει τα διανύσματά εισόδου x_i σε M(2^N επίπεδα κβάντισης) συστάδες ($Q = \{Q_1, \ldots, Q_M\}$), σύμφωνα με την απόσταση τους (2.5) από κάποια αντιπροσωπευτικά μ_i διανύσματα (ένα για κάθε συστάδα), που ξανάυπολογίζονται σε κάθε επανάληψη (2.6).

$$\sum_{i=1}^{N} \min_{\mu_j \in Q} \|x_i - \mu_j\|_2^2 \tag{2.5}$$

$$\mu_i^{(t+1)} = \frac{1}{|Q_i|} \sum_{x_j \in Q_i^{(t)}} x_j \tag{2.6}$$

Τα αρχικά αντιπροσωπευτικά διανύσματα επιλέγονται τυχαία. Στην περίπτωση της κβάντισης, τα αντιπροσωπευτικά διανύσματα, θεωρούνται ως οι τιμές των επιπέδων κβάντισης και οι συστάδες ως τα όρια των ζωνών κβάντισης. Σύμφωνα με αυτά, η υλοποίηση φαίνεται στο 3.10. Αρχικά, υπολογίζεται ο αριθμός των επιπέδων κβάντισης (γραμμή 5) και το σήμα εισόδου περιορίζεται στης ελάχιστες και μέγιστες τιμές που έχουν ορισθεί (γραμμές 8-12). Επόμενο βήμα είναι η μετατροπή του σήματος εισόδου σε διανύσματα $x \in R^2$. Αυτό επιτυγχάνεται με την χρήση της εντολής reshape της MATLAB (γραμμές 14-18). Έπειτα, εκτελείται στον παραγόμενο πίνακα των δισδιάστατων διανυσμάτων ο αλγόριθμος K-means με 2^N συστάδες (γραμμή 21). Τέλος, υπολογίζεται η μέση παραμόρφωση. Χρησιμοποιούνται οι τύποι 2.2 και 2.3, με την μόνη διαφορά ότι επειδή πλέον έχουμε διανύσματα, για την διαφορά χρησιμοποιείται η ευκλείδεια απόσταση (γραμμές 23-31).

2.1.3 Υπολογισμός SQNR - MSE

Σε αυτό το κεφάλαιο γίνεται αναφορά στην υλοποίηση για τον υπολογισμό του Signal to Quantization Noise Ratio (SQNR) και του Mean Squared Error (MSE) για τον βαθμωτό και διανυσματικό κβαντιστή.

Μη Ομοιόμορφος Βαθμωτός Κβαντιστής

Χρησιμοποιήθηκε μία συνάρτηση με όνομα sqnr_mse_calculation_scalar, που δέχεται ως είσοδο το αρχικό σήμα, το πίνακα με τους δείκτες επιπέδων κβάντισης που παράγεται από τον μη ομοιόμορφο βαθμωτό κβαντιστή, τα επίπεδα κβάντισης του μη ομοιόμορφου βαθμωτού κβαντιστή, καθώς και το πλήθος τους. Επιστρέφει το SQNR σε dB και το MSE. Η υλοποίσηση φαίνεται στο 3.11. Αρχικά, κατασκευάζεται το κβαντισμένο σήμα (γραμμές 6-12) και στη συνέχεια υπολογίζεται το $E\left[X^2\right]$, και το $E\left[\tilde{X}^2\right]$ (γραμμές 14-19), όπου το X είναι το αρχικό σήμα και το $\tilde{X}$ είναι το κβαντισμένο σήμα. Σύμφωνα με την θεωρία το $E\left[\tilde{X}^2\right]$ αποτελεί το Mean Squared Error(γραμμές 24-25). Το SQNR υπολογίζεται σύμφωνα με τον τύπο 2.7 (γραμμές 21-22).

$$SQNR = \frac{E\left[X^2\right]}{E\left[\tilde{X}^2\right]} = \frac{P_X}{P_{\tilde{X}}} \tag{2.7}$$

Όπου το E[ ] δηλώνει την μέση τιμή ενός σήματος και το P την ισχύ ενός σήματος. Για να μετατραπεί το SQNR σε dB χρησιμοποιείται ο τύπος 2.8.

$$SQNR\_in\_dB = 10\log_{10}(SQNR) \tag{2.8}$$

Μη Ομοιόμορφος Διανυσματικός Κβαντιστής

Για τον υπολογισμό του SQNR και του MSE για τον διανυσματικό κβαντιστή χρησιμοποιήθηκε μία παρόμοια συνάρτηση με την sqnr_mse_calculation_scalar, που ονομάζεται sqnr_mse_calculation_vector. Δέχεται ως είσοδο το σήμα με τα δισδιάστατα διανύσματα που χρησιμοποιήθηκε για τον αλγόριθμο K-means, το πίνακα με τους δείκτες επιπέδων κβάντισης που παράγεται από τον μη ομοιόμορφο διανυσματικό κβαντιστή και τα επίπεδα κβάντισης του μη ομοιόμορφου διανυσματικού κβαντιστή. Η υλοποίηση φαίνεται στο 3.12. Αρχικά, κατασκευάζεται το κβαντισμένο σήμα (γραμμές 5-6) και έπειτα υπολογίζονται τα $E\left[X^2\right]$ και $E\left[\tilde{X}^2\right]$ (γραμμές 8-13), λαμβάνοντας υπόψιν ότι τα σήματα αποτελούνται από δισδιάστατα διανύσματα. Όπως και στην περίπτωση του μη ομοιόμορφου βαθμωτού κβαντιστή το MSE ισούται με το $E\left[\tilde{X}^2\right]$ (γραμμές 18-19) και το SQNR υπολογίζεται από τον τύπο 2.7 και μετατρέπεται σε dB με τον τύπο 2.8 (γραμμές 15-16).

Κβαντιστής	Bits/Έξοδο	SQNR	MSE	Μέση Παραμόρφωση
Βαθμωτός	2	4.979123	0.316311	0.3163
Βαθμωτός	3	12.888236	0.051192	0.0512
Βαθμωτός	4	19.670539	0.010739	0.0107
Διανυσματικός	4	9.797554	0.104277	0.2022
Διανυσματικός	6	15.496751	0.028072	0.1050
Διανυσματικός	8	22.107848	0.006126	0.0495

Table 2.1: Results of the quantizers.

2.1.4 Ζητούμενο 1

Ολοκληρώνοντας τα παραπάνω, είμαστε σε θέση να περάσουμε στην περιγραφή της υλοποίησης των ζητούμενων. Ξεκινώντας με το ζητούμενο 1, η υλοποίηση φαίνεται στο 3.13. Αρχικά, παράγεται το σήμα που αντιστοιχεί στην πρώτη πηγή (γραμμές 5-9) σύμφωνα με τις προδιαγραφές που ζητούνται, και στην συνέχεια υπολογίζονται η ελάχιστη και η μέγιστη επιτρεπόμενη τιμή (γραμμές 11-13). Στο πλαίσιο των πειραμάτων που εκτελέστηκαν χρησιμοποιήθηκαν η ελάχιστη και η μέγιστη τιμή του παραγόμενου σήματος, ελαττωμένες κατά ένα πολύ μικρό αριθμό, ως ελάχιστες και μέγιστες επιτρεπόμενες τιμές, αλλά είναι δυνατόν να χρησιμοποιηθούν οποιεσδήποτε τιμές. Επόμενο βήμα είναι ο ορισμός του αριθμού των bits που θα χρησιμοποιηθούν για κάθε κβαντιστή (γραμμές 15-19). Για τον βαθμωτό χρησιμοποιούνται 2,3 και 4 bits, ενώ για τον διανυσματικό 4,6 και 8 bits. Για τον διανυσματικό χρησιμοποιήθηκαν διπλάσια bits ανά έξοδο σε σχέση με τον βαθμωτό ώστε να έχουν ίδια bits ανά δείγμα, αφού ο διανυσματικός κβαντιστής είναι δύο διαστάσεων.

Ακολουθεί η χρήση των δύο κβαντιστών, που έχουν αναλυθεί στα προηγούμενα κεφάλαια, και ο υπολογισμός των SQNR και MSE για αυτούς (γραμμές 21-57). Συνολικά εκτελούνται 6 πειράματα, 3 με κάθε κβαντιστή, για κάθε τιμή των αντίστοιχων bits ανά δείγμα. Τα αποτελέσματα φαίνονται στον πίνακα 2.1. Για τον βαθμωτό κβαντιστή συμπεριλήφθηκε μόνο η τελική τιμή της μέσης παραμόρφωσης.

Αρχικά, παρατηρούμε ότι με την αύξηση των bits ανά δείγμα κατά 1, το SQNR αυξάνεται κατά 6-7 dB το οποίο είναι αναμενόμενο. Η μέση παραμόρφωση για κάθε κβαντιστή μειώνεται, με την αύξηση των bits ανά δείγμα, το οποίο είναι επίσης αναμενόμενο αφού η διαδικασία της κβάντισης γίνεται όλο και πιο λεπτομερή με την αύξηση των επιπέδων κβάντισης. Η μέση παραμόρφωση του διανυσματικού είναι μικρότερη σε σχέση με τον βαθμωτό σε όλες τις αντίστοιχες περιπτώσεις. Τέλος, και για τους δύο κβαντιστές παρατηρούμε μείωση στο MSE καθώς αυξάνονται τα επίπεδα κβάντισης, που όπως αναφέρθηκε και πιο πριν, οφείλεται ότι στο γεγονός ότι η κβάντιση γίνεται όλο και πιο λεπτομερή, με την αύξησή των σταθμών κβάντισης.

2.1.5 Ζητούμενο 2

Για το ζητούμενο 2, υλοποιήθηκαν δύο αρχεία κώδικα MATLAB. Το πρώτο περιέχει των κώδικα για το υποερώτημα α) και βρίσκεται στο 3.14. Η διαδικασία που ακολουθήθηκε είναι εξαιρετικά απλή. Αρχικά, δημιουργείται το αρχικό σήμα σύμφωνα με τις προδιαγραφές που δίνονται (γραμμές 3-14) και έπειτα υπολογίζονται η ελάχιστη και η μέγιστη επιτρεπόμενη τιμή (γραμμές 16-18), όπως και στο ζητούμενο 1. Στην συνέχεια ορίζονται ο αριθμός των bits ανά έξοδο και το κατώφλι epsilon που θα χρησιμοποιηθεί για τον βαθμωτό κβαντιστή (γραμμές 20-22). Έπειτα, για τις διάφορες τιμές των bits ανά έξοδο, εκτελείται ο βαθμωτός κβαντιστής και δημιουργείται ένα διάγραμμα που δείχνει πως εξελίσσονται οι τιμές του SQNR σε σχέση με τον αριθμό των επαναλήψεων K_{max} του βαθμωτού κβαντιστή (γραμμές 24-55). Για 2 bits ανά έξοδο το διάγραμμα που παράχθηκε είναι το 2.1, για 3 bits ανά έξοδο το 2.2 και για 4 bits ανά έξοδο το 2.3.

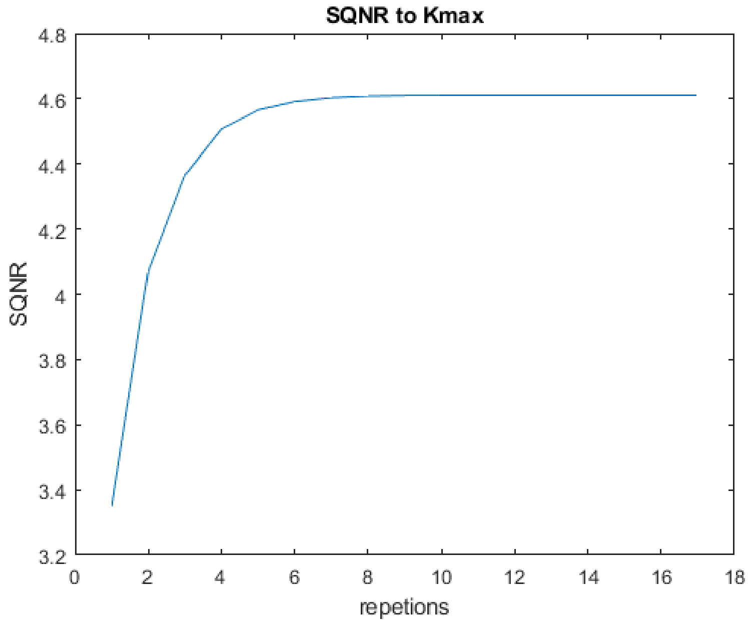

Figure 2.1: SQNR - K_{max} comparison for 2 bits per output

Για τα άλλα δύο υποερωτήματα ακολουθείτε παρόμοια διαδικασία με το ζητούμενο 1. Ο κώδικας φαίνεται στο 3.15. Αρχικά, παράγεται το αρχικό σήμα σύμφωνα με τις προδιαγραφές που δίνονται (γραμμές 5-16) και έπειτα υπολογίζονται η ελάχιστη και η μέγιστη επιτρεπόμενη τιμή (γραμμές 18-20), όπως και στο ζητούμενο 1. Στην συνέχεια ορίζονται τα δύο διανύσματα που περιέχουν τις τιμές των bits ανά έξοδο (γραμμές 22-26) και το αρχικό σήμα κβαντίζεται με τους δύο κβαντιστές. Συνολικά εκτελούνται 6 πειράματά, 3 με κάθε κβαντιστή, για κάθε τιμή των αντίστοιχων bits ανά δείγμα. Τέλος, για κάθε πείραμα υπολογίζονται το SQNR και το MSE. Τα αποτελέσματα

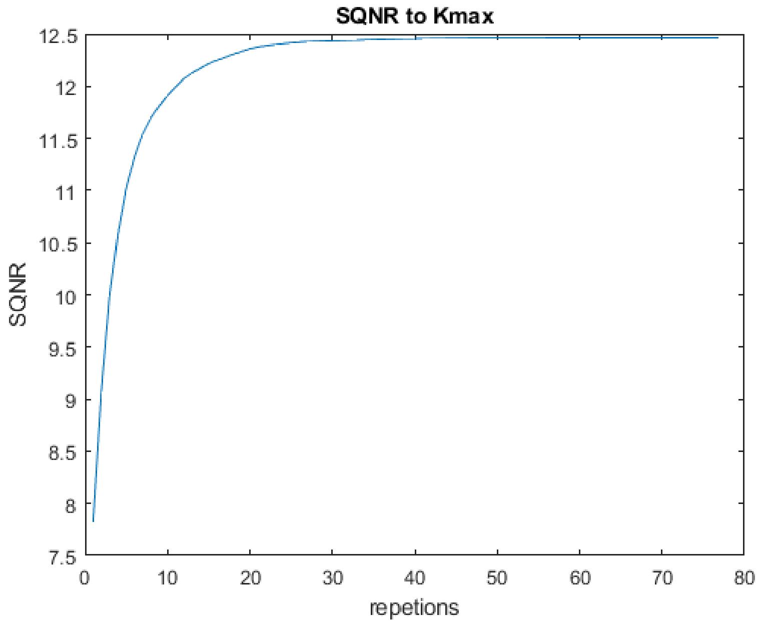

Figure 2.2: SQNR - K_{max} comparison for 3 bits per output

Κβαντιστής	Bits/Έξοδο	SQNR	MSE	Μέση Παραμόρφωση
Βαθμωτός	2	4.969620	0.405198	0.4052
Βαθμωτός	3	12.642560	0.069243	0.0692
Βαθμωτός	4	18.717992	0.017093	0.0171
Διανυσματικός	4	9.694242	0.136503	0.2317
Διανυσματικός	6	15.448419	0.036285	0.1194
Διανυσματικός	8	22.051790	0.007932	0.0558

Table 2.2: Results of the quantizers.

φαίνονται στον πίνακα 2.2.

Τα συμπεράσματα που απορρέουν από τα αποτελέσματα είναι παρόμοια με αυτά για την πρώτη πηγή. Όπως και στο ζητούμενο 1, καθώς αυξάνονται οι τιμές των bits ανά δείγμα αυξάνεται και το SQNR με ρυθμό 6db ανά bit. Μία σημαντική διαφορά είναι ότι ο βαθμωτός κβαντιστής πετυχαίνει μικρότερη μέση παραμόρφωση για τιμές 3 και 4 bits ανά δείγμα. Όπως και στο ζητούμενο 1 παρατηρείται ότι το μέσο τετραγωνικό σφάλμα (MSE) μειώνεται ραγδαία με την αύξηση των επιπέδων κβάντισης. Αυτό για άλλη μία φορά οφείλεται στο γεγονός ότι όσο πιο πολλές στάθμες κβάντισης χρησιμοποιούμε τόσο πιο λεπτομερή και καλύτερη γίνεται η διαδικασία της κβάντισης. Αυτό έχει ωστόσο ως αρνητικό την αύξηση των απαιτήσεων όσο αφορά τη υπολογιστική δύναμη που χρειάζεται για την κβάντιση του σήματος, καθώς και την αύξηση του απαιτούμενο αποθηκευτικού χώρου.

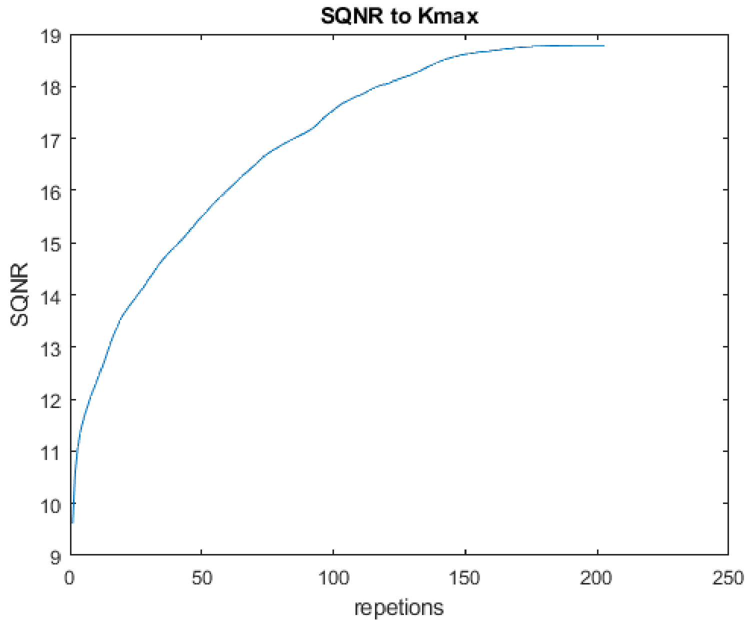

Figure 2.3: SQNR - K_{max} comparison for 4 bits per output

*Το παρόν μάθημα αποτελεί ένα από τα τελευταία μαθήματα μου για πτυχίο, μαζί με αυτό των πολυδιάστατων δομών δεδομένων, για το λόγο αυτό θα παρακαλούσα για την όσο το δυνατό επιείκεια σας. Σας ευχαριστώ πολύ.

Chapter 3

Κώδικες Υλοποίησης MATLAB

Κώδικες MATLAB

Παρακάτω ακολουθούν οι κώδικες για την υλοποίηση κάθε ερωτήματος.

Κωδικοποίηση Huffman

```
function [dict ,avglen] = huffmandict(sym, prob)
%Generates a dictionary based on huffman code

%Get the length of the vectors(they are the same)
n = length(prob);

%Sort the probabilities
[p,ind] = sort(prob);

%Combinations matrix
m=zeros(n-1,n);


%Combining the elements. We put the combinded
%element always first
for i=1:n-1
    m(i,:)=[ind(1:n-i+1),zeros(1,i-1)];
    p=[p(1)+p(2),p(3:n),1];
    [p,ind]=sort(p);
end


%The prefix binary tree
tree = repmat(' ',n-1,n^2);

%Initializing the tree
tree(1, n) = '0';
tree(1,n*2) = '1';


%Constructing the tree
for i=2:n-1
    comb_sym_ind = (find(m(n+1-i,:)==1));
    tree(i,1:n-1) = tree(i-1,n*(comb_sym_ind-1)+2:n*(comb_sym_ind));
    tree(i,n)='0';
    tree(i,n+1:2*n-1) = tree(i-1,n*(comb_sym_ind-1)+2:n*(comb_sym_ind))
    ;
    tree(i,2*n)='1';
```

```
    for j=1:i-1
        other_sym_ind = (find(m(n-i+1,:)==j+1));
        tree(i,(j+1)*n+1:(j+2)*n)=tree(i-1,n*(other_sym_ind-1)+1:n*other_sym_ind);
    end
end

%Extract the codes for each symbol
for i=1:n
    symbol_ind = (find(m(1,:)==i));
    huff_code(i,1:n)= tree(n-1,n*(symbol_ind-1)+1:n*symbol_ind);
end

%Remove some whitespaces
huff_code = strtrim(huff_code);

%Creating the dictionary
dict = cell(n+1,2);

dict{1,1} = {'Symbols'};
dict{1,2} = {'Code'};


for j=2:n+1
    dict{j,1} = sym(:,j-1);
    dict{j,2} = huff_code(j-1,:);
end

%Calculate average length
for i=1:n
    len(i)=length(find(abs(huff_code(i,:))~=32));
    lp(i) = len(i)*prob(i);
end

avglen = sum(lp);
```

Αλγόριθμος 3.1: The matlab code of huffmandict function

```
function code = huffmanenco(sig, dict)
%Encodes the input based on a huffman dictionary

%Get the length of the dictionary and signal
dictionary_len = sum(cellfun('size',dict,1));
dict_len = dictionary_len(2);
sig_len = numel(sig);

%Get the symbols from the dictionary
symbols = [dict{2:dict_len,1}];

%For extension
if dictionary_len(1)>dictionary_len(2)
    %Initialize the code
    code = cell(sig_len/2,1);

    %counter
    k=1;

    %Match Dictionary Symbols with Signal Symbols and create the encoding
    for i = 1:sig_len/2
        indx_a = strfind(symbols(1,:),sig(k));
        indx_b = strfind(symbols(2,:),sig(k+1));
        indx=intersect(indx_a,indx_b);
        code{i} = dict{indx+1,2};
        k=k+2;
    end
else

    %For regular source
    %Initialize the code
    code = cell(sig_len,1);
```

```

    %Match Dictionary Symbols with Signal Symbols and create the encoding
    for i = 1:sig_len
        indx = strfind(symbols,sig(i));
        code{i} = dict{indx+1,2};
    end
end
```

Αλγόριθμος 3.2: The matlab code of huffmanenco function

```
function sig = huffmandeco(code, dict)
%Decodes the input based on a huffman dictionary

%Get the length of the dictionary and code
dict_len = sum(cellfun('size',dict,1));
dict_len = dict_len(2);
code_len = size(code,1);

%Get the symbols code from the dictionary
symbols_code = cell(dict_len,1);
for i = 2:dict_len
    symbols_code{i} = dict{i,2};
end

%Initialize the signal
sig = [];
sig_temp =[];

%Match Dictionary Symbols with Code Symbols and create the decoding
for i = 1:code_len
    for j=1:dict_len
        if strcmp(symbols_code(j),code{i})
            sig_temp =[sig_temp, dict{j,1}];
        end
    end
end
sig = transpose(sig_temp);


%For extension
if size(sig,2)>1
    %Get final text string
    temp =[];

    for i=1:size(sig,1)
        temp = [temp sig(i,1), sig(i,2)];
    end
    sig = temp;
else
    %Get final string
    sig = convertCharsToStrings(sig);
end
```

Αλγόριθμος 3.3: The matlab code of huffmandeco function

```
function [stats] = huffmanstats(prob,avglen)
%Calculates stats about the encoding

%Calculates Entropy
Entropy = prob*log2(1./prob)';

%Calculates the effiency of the endoing
eff = Entropy/avglen;

%Get stats
stats = table(Entropy,avglen,eff,'VariableNames',{'Entropy' 'Average
    Length' 'efficiency'});
```

Αλγόριθμος 3.4: The matlab code of huffmanstats function

```
%Input the file
text = fileread('cvxopt.txt');

%Count each letter
number_text = lower(text) - '0' - 48;
edges = 1:27;

%count(1) -> a, ...., count(26) --> z
counts = histcounts(number_text, edges);

%Count whitespaces
num_of_spaces = 6184 - sum(counts);

counts = [counts,num_of_spaces];

%Compute the probabilities of each letter and whitespace
prob = counts./6184;

if sum(prob) ~= 1
    error('Sum of probabilities must be equal to 1')
end

%Create the symbol array
symbols = 'a':'z';
symbols = [symbols, ' '];

%Create the huffman dictionary
[dict ,avglen] = huffmandict(symbols, prob);

%Encode the text
code = huffmanenco(text, dict);

%Decode the code
sig = huffmandeco(code, dict);

if ~isequal(text,sig)
    error('The decoded input is different from the original input')
end

%Get stats
stats = huffmanstats(prob,avglen);
```

Αλγόριθμος 3.5: The matlab code of encoding/decoding the first source

```
%Input the file
text = fileread('cvxopt.txt');

%Input English frequencies
p = fileread('frequencies.txt');
p = strsplit(p);

%Initialize and parse english frequencies
prob = [];

for i=2:2:size(p,2)
    prob = [prob,str2double(p{i})];
end

%Create the symbol array
symbols = 'a':'z';
symbols = [symbols, ' '];

%Create the huffman dictionary
[dict ,avglen] = huffmandict(symbols, prob);

```

```
%Encode the text
code = huffmanenco(text, dict);

%Decode the code
sig = huffmandeco(code, dict);

if ~isequal(text,sig)
    error('The decoded input is different from the original input')
end

%Get stats
stats = huffmanstats(prob,avglen);
```

Αλγόριθμος 3.6: The matlab code of encoding/decoding the first source with english probabilities

```
%Input the file
text = fileread('cvxopt.txt');

%Count each letter
number_text = lower(text) - '0' - 48;
edges = 1:27;

%count(1) -> a, ...., count(26) --> z
counts = histcounts(number_text, edges);

%Count whitespaces
num_of_spaces = 6184 - sum(counts);

counts = [counts,num_of_spaces];

%Compute the probabilities of each letter and whitespace
prob = counts./6184;

if sum(prob) ~= 1
    error('Sum of probabilities must be equal to 1')
end

%Create the symbol array
symbols = 'a':'z';
symbols = [symbols, ' '];

sym = combvec(symbols,symbols);
p = combvec(prob,prob);

sym = char(sym);

final_prob = [];
final_prob=double(final_prob);
final_prob = p(1,:).*p(2,:);

%Create the huffman dictionary
[dict ,avglen] = huffmandict(sym, final_prob);

%Encode the text
code = huffmanenco(text, dict);

%Decode the code
sig = huffmandeco(code, dict);

if ~isequal(text,sig)
    error('The decoded input is different from the original input')
end

%Get stats
stats = huffmanstats(final_prob ,avglen);
```

Αλγόριθμος 3.7: The matlab code of encoding/decoding the first source using second extension

```
%Load and parse the cameraman picture
cameraman = load('cameraman.mat');
cameraman_values = cameraman.i;
cameraman_len = length(cameraman_values);

%Transform the cameraman matrix into a vector using reshape
cameraman_vec = reshape(cameraman_values,cameraman_len^2,1);

%Guess the probabilities
[counts,unique] = groupcounts(cameraman_vec);

cameraman_vec_size = length(cameraman_vec);

prob = [];

for i=1:length(counts)
    prob(i) = counts(i)/cameraman_vec_size;
end

%Huffman Encoding
%Create the Dictionary
unique = transpose(unique);
[dict ,avglen] = huffmandict(unique, prob);

%Encode the source
code = huffmanenco(cameraman_vec, dict);

%Convert to array
code_array = cell2mat(code(:,:));

%Remove some whitespaces
code_array = strtrim(code_array);

%Convert Binary to numbers
number_array = code_array-'0';

%reshape the array
number_array = reshape(number_array, 1,65536*16);

%remove the whitespaces
number_array = number_array(number_array~=-16);

%Transmitting the signal
y = bsc(number_array);

%Count the errors
err_cnt = 0;
not_err_cnt =0;

for i=1:length(number_array)
    if number_array(i)~=y(i)
        err_cnt = err_cnt + 1;
    end
    if number_array(i)==y(i)
        not_err_cnt = not_err_cnt + 1;
    end
end

%Calculate the probability p
p_error = err_cnt/length(number_array);
p_correct = not_err_cnt/length(number_array);

%Verify it is correct
if p_error==1-p_correct
    error('Probabilities arent correct');
end

%Calculate the capacity of the channel
H = -p_correct * log2(p_correct) - (1-p_correct)*log2(1-p_correct);
Capacity = 1 - H;
```

Αλγόριθμος 3.8: The matlab code of encoding/decoding the second source

Κωδικοποίηση PCM

```
function [xq, centers, D_scalar, sqnr_vec] = Lloyd_Max(x, N, min_value,
    max_value,epsilon)
%Lloyd Max scalar quantizer
%    Inputs:
%         x = original signal
%         N = number of bits
%         min_value = minimum value for the outuputs of the signal
%         max_value = maximum value for the outuputs of the signal
          epsilon = defines the number of repetitions
%    Outuputs:
%         xq = quantized output signal
%         centers = the quantization centers
%         D = average distortion

%Step 0: Initialize variables and preprocess the signal
D=[-1 1];
k=2;
levels = 2^N;
minv = min_value;
maxv = max_value;
x(x<minv) = minv;
x(x>maxv) = maxv;
if epsilon ==0
    epsilon = eps;
end


%Step 1: Guess the initial reprsentitive levels
initial_levels = (maxv-minv).*rand(levels-2,1) + minv;
centers = [minv transpose(initial_levels) maxv];

centers = sort(centers);

%Step 2: As long as the algorithm doesn't converge repeat the steps:
%    2a: Calculate the decision thresholds
%    2b: Calculate the quantized signal and the median distortion
%    2c: Calculate the new reprsentitives levels

while abs(D(k)-D(k-1))>= epsilon

    %The quantized signal
    xq=[];

    %The decision thresholds
    T=[];


    %Step 2a
    for i=1:length(centers)-1
        T(i) = (centers(i) + centers(i+1))/2;
    end
    T = [minv T maxv];


    %Step 2b:
    %Quantized Signal Calculation
    for i=1:size(x)
        for j = 1:length(T)-1
            if x(i)>= T(j) && x(i) <= T(j+1)
                xq(i) = levels+1-j;
            end
        end
    end
    xq = transpose(xq);

    %Distortion Calculation and calculation of total value of signal
    %x in every level and counts the number of values in each level
    Distortion = 0;
```

```
    sumx = zeros ( 1 , length (centers)-1);
    countx= zeros ( 1 , length (centers)-1);
    for i=1:size(x)
        for j = 1:length(T)-1
            if x(i)>= T(j) && x(i) <= T(j+1)
                Distortion = Distortion + (x(i) - centers(j))^2;
                sumx(j) = sumx(j)+ x(i);
                countx(j) = countx(j) + 1;
            end
        end
    end

    Average_Distortion = Distortion/length(x);
    D = [D Average_Distortion];
    D_scalar = D;
    k=k+1;

    %Step 2c:
    %Calculate the new reprsentitives levels
    for i=2:length(centers)-1
        centers(i) = sumx(i)/countx(i);
        if isnan(centers(i))
            centers(i) = centers(i-1) + 0.01;
        end
    end

    %For 2a)
    sqnr_vec(k-2) = sqnr_mse_calculation_scalar(x, xq, centers, N);

end
```

Αλγόριθμος 3.9: The matlab code of the lloyd-max quantizer

```
function [idx, C, D_vec, Vector_matrix] = Vector_Quantizer_Kmeans(x, N,
    min_value, max_value)
%Vector quantizer based on K-means clustering

%levels of the quantization process = number of centroids used for
    Kmeans
levels = 2^N;

%Filtering the Values greater and less than min and max values given
minv = min_value;
maxv = max_value;

x(x<minv) = minv;
x(x>maxv) = maxv;

%Reshape input signal to matrix of vectors
sig_size = length(x);
matrix_dimension_size = sig_size/2;

Vector_matrix = reshape(x,matrix_dimension_size,2);

%Run Kmeans Algorithm
[idx,C] = kmeans(Vector_matrix, levels);

%Average Distortion Calculation
Total_Distortion = 0;

for i=1:length(Vector_matrix)
    representative_index = idx(i);
    Total_Distortion = Total_Distortion + norm(Vector_matrix(i,:)-C(
        representative_index,:), 2);
end

D_vec = Total_Distortion/length(x);
```

Αλγόριθμος 3.10: The matlab code of the Vector Quantizer based on Kmeans

```
function [sqnr,MSE] = sqnr_mse_calculation_scalar(signal, qindices, qcenters,N)
%SQNR Calculation --> SQNR = P_signal_x / P_quantized_x = E[X^2] / E[~X^2]
%where E[X^2] = mean(signal^2) and E[~X^2] = mean((x - ~x)^2) = mean((x - Q(x))^2) = MSE
levels = 2^N;

%Quantized signal
for j=1:length(qindices)
    final_qindices(j) = levels + 1 - qindices(j);
end

quantized_signal = qcenters(final_qindices);
quantized_signal = transpose(quantized_signal);

%E[X^2] = mean(signal^2)
E_X = mean(signal.^2);

%E[~X^2] = mean((x - ~x)^2)
temp = signal - quantized_signal;
E_X_tilde = mean(temp.^2);

%sqnr calculation in db
sqnr = 10*log10(E_X/E_X_tilde);

%Mean Squared Error
MSE = E_X_tilde;
```

Αλγόριθμος 3.11: The matlab code for the scalar sqnr - mse calculation

```
function [sqnr,MSE] = sqnr_mse_calculation_vector(signal, qindices, qcenters)
%SQNR Calculation --> SQNR = P_signal_x / P_quantized_x = E[X^2] / E[~X^2]
%where E[X^2] = mean(signal^2) and E[~X^2] = mean((x - ~x)^2) = mean((x - Q(x))^2) = MSE

%Quantized signal
quantized_signal = qcenters(qindices,:);

%E[X^2] = mean(signal^2)
E_X = sum(mean(signal.^2))/2;

%E[~X^2] = mean((x - ~x)^2)
temp = signal - quantized_signal;
E_X_tilde = sum(mean(temp.^2))/2;

%sqnr calculation in db
sqnr = 10*log10(E_X/E_X_tilde);

%Mean Squared Error
MSE = E_X_tilde;
```

Αλγόριθμος 3.12: The matlab code for the vector sqnr - mse calculation

```
%Generate the required signal and run the lloyd max scalar quantizer and
%the vector quantizer based on k-means. After that compare the two
%implementations.

%Size of the signal
M = 10000;

%The signal itself
signal_x = randn(M,1);
```

```

%Min and Max values allowed
min_value = min(signal_x)+0.1;
max_value = max(signal_x)-0.1;

%Levels for each quantizer. Since we quantize two inputs at the same time
%for the vector quantizer, it will be 2*N for him. This is done to compare
%the two quantizers more efficiently.
N_scalar = [2,3,4];
N_vector = [4,6,8];

%First round for scalar N = 2 and vector N = 4.
x = signal_x;
[xq, centers, D_scalar1] = Lloyd_Max(x, N_scalar(1), min_value, max_value, 0);

y = signal_x;
[idx, C, D_vec1, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(1), min_value, max_value);

%SQNR Calculation
sqnr_scalar = [];
sqnr_vector = [];
MSE_scalar = [];
MSE_vector = [];

[sqnr_scalar(1), MSE_scalar(1)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(1));
[sqnr_vector(1), MSE_vector(1)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

%Second round for scalar N = 3 and vector N = 6.
x = signal_x;
[xq, centers, D_scalar2] = Lloyd_Max(x, N_scalar(2), min_value, max_value, 0);

y = signal_x;
[idx, C, D_vec2, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(2), min_value, max_value);

%SQNR Calculation
[sqnr_scalar(2), MSE_scalar(2)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(2));
[sqnr_vector(2), MSE_vector(2)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

%Third round for scalar N = 4 and vector N = 8.
x = signal_x;
[xq, centers, D_scalar3] = Lloyd_Max(x, N_scalar(3), min_value, max_value, 0);

y = signal_x;
[idx, C, D_vec3, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(3), min_value, max_value);

%SQNR Calculation
[sqnr_scalar(3), MSE_scalar(3)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(3));
[sqnr_vector(3), MSE_vector(3)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

fprintf('For N = 2 bits the scalar SQNR was %f and the MSE was %f, while for N=4 the vector SQNR was %f and the MSE was %f.\n\n', sqnr_scalar(1), MSE_scalar(1),sqnr_vector(1), MSE_vector(1))

fprintf('For N = 3 bits the scalar SQNR was %f and the MSE was %f, while for N=6 the vector SQNR was %f and the MSE was %f.\n\n', sqnr_scalar(2), MSE_scalar(2),sqnr_vector(2), MSE_vector(2))

fprintf('For N = 4 bits the scalar SQNR was %f and the MSE was %f,
```

```
    while for N=8 the vector SQNR was %f and the MSE was %f.\n\n',
    sqnr_scalar(3), MSE_scalar(3),sqnr_vector(3), MSE_vector(3))
```

Αλγόριθμος 3.13: The matlab code for the quantization of the first source

```
%Generate the required signal and run the lloyd max scalar quantizer.

%Size of the signal
M = 10000;

%The white noise
signal_x = randn(M,1);

%The a,b coefficients for the filter
b = 1;
a = [1 1/2 1/3 1/4 1/5 1/6 ];

%The filtered signal
signal_y = filter(b,a,signal_x);

%Min and Max values allowed
min_value = min(signal_y)+0.1;
max_value = max(signal_y)-0.1;

%Levels for the scalar quantizer.
N_scalar = [2,3,4];
epsilon = 10^(-16);

%First round for scalar N = 2 and epsilon = 10^(-16).
x = signal_x;
[xq, centers, D_scalar1, sqnr_vec1] = Lloyd_Max(x, N_scalar(1),
    min_value, max_value, epsilon);

%Plot the SQNR to Kmax figure
figure = figure();
plot(1:length(sqnr_vec1),sqnr_vec1)
title('SQNR to Kmax')
xlabel('repetions' )
ylabel('SQNR')

%Second round for scalar N = 3 and epsilon = 10^(-16).
x = signal_x;
[xq, centers, D_scalar3, sqnr_vec2] = Lloyd_Max(x, N_scalar(2),
    min_value, max_value, epsilon);

%Plot the SQNR to Kmax figure
figure = figure();
plot(1:length(sqnr_vec2),sqnr_vec2)
title('SQNR to Kmax')
xlabel('repetions' )
ylabel('SQNR')

%Third round for scalar N = 4 and epsilon = 10^(-16).
x = signal_x;
[xq, centers, D_scalar5, sqnr_vec3] = Lloyd_Max(x, N_scalar(3),
    min_value, max_value, epsilon);

%Plot the SQNR to Kmax figure
figure = figure();
plot(1:length(sqnr_vec3),sqnr_vec3)
title('SQNR to Kmax')
xlabel('repetions' )
ylabel('SQNR')
```

Αλγόριθμος 3.14: The matlab code for comparison of SQNR based on the Kmax repetitions.

```
%Generate the required signal and run the lloyd max scalar quantizer and
%the vector quantizer based on k-means. After that compare the two
%implementations.

%Size of the signal
M = 10000;

%The white noise
signal_x = randn(M,1);

%The a,b coefficients for the filter
b = 1;
a = [1 1/2 1/3 1/4 1/5 1/6 ];

%The filtered signal
signal_y = filter(b,a,signal_x);

%Min and Max values allowed
min_value = min(signal_y)+0.1;
max_value = max(signal_y)-0.1;

%Levels for each quantizer. Since we quantize two inputs at the same time
%for the vector quantizer, it will be 2*N for him. This is done to compare
%the two quantizers more efficiently.
N_scalar = [2,3,4];
N_vector = [4,6,8];

%First round for scalar N = 2 and vector N = 4.
x = signal_y;
[xq, centers, D_scalar1] = Lloyd_Max(x, N_scalar(1), min_value, max_value, 0);

y = signal_y;
[idx, C, D_vec1, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(1), min_value, max_value);

%SQNR Calculation
sqnr_scalar = [];
sqnr_vector = [];
MSE_scalar = [];
MSE_vector = [];

[sqnr_scalar(1), MSE_scalar(1)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(1));
[sqnr_vector(1), MSE_vector(1)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

%Second round for scalar N = 3 and vector N = 6.
x = signal_y;
[xq, centers, D_scalar2] = Lloyd_Max(x, N_scalar(2), min_value, max_value, 0);

y = signal_y;
[idx, C, D_vec2, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(2), min_value, max_value);

%SQNR Calculation
[sqnr_scalar(2), MSE_scalar(2)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(2));
[sqnr_vector(2), MSE_vector(2)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

%Third round for scalar N = 4 and vector N = 8.
x = signal_y;
[xq, centers, D_scalar3] = Lloyd_Max(x, N_scalar(3), min_value, max_value, 0);

y = signal_y;
[idx, C, D_vec3, vec_mat] = Vector_Quantizer_Kmeans(y, N_vector(3),
```

```
    min_value, max_value);

%SQNR Calculation
[sqnr_scalar(3), MSE_scalar(3)] = sqnr_mse_calculation_scalar(x, xq, centers, N_scalar(3));
[sqnr_vector(3), MSE_vector(3)] = sqnr_mse_calculation_vector(vec_mat, idx, C);

fprintf('For N = 2 bits the scalar SQNR was %f and the MSE was %f, while for N=4 the vector SQNR was %f and the MSE was %f.\n\n', sqnr_scalar(1), MSE_scalar(1),sqnr_vector(1), MSE_vector(1))

fprintf('For N = 3 bits the scalar SQNR was %f and the MSE was %f, while for N=6 the vector SQNR was %f and the MSE was %f.\n\n', sqnr_scalar(2), MSE_scalar(2),sqnr_vector(2), MSE_vector(2))

fprintf('For N = 4 bits the scalar SQNR was %f and the MSE was %f, while for N=8 the vector SQNR was %f and the MSE was %f.\n\n', sqnr_scalar(3), MSE_scalar(3),sqnr_vector(3), MSE_vector(3))
```

Αλγόριθμος 3.15: The matlab code for the quantization of the second source.