Πανεπιστήμιο Πατρών
Τμήμα Μηχ. Η/Υ & Πληροφορικής

Ψ Η Φ Ι Α Κ Ε Σ Τ Η Λ Ε Π Ι Κ Ο Ι Ν Ω Ν Ι Ε Σ

Εργαστηριακές Ασκήσεις 2020-2021 : 1ο σετ

Γάτου Παρασκευή
ΑΜ: 1054370

Ο κώδικας για το κάθε ερώτημα ξεχωριστά βρίσκεται στο τέλος του αρχείου!

Ερώτημα 1 – Κωδικοποίηση Huffman

1. Στο περιβάλλον Matlab υλοποιήθηκαν οι εξής συναρτήσεις:
   - **huffmandict(alphabet , probabilitymatrix)**: δέχεται ως είσοδο το αλφάβητο και τις πιθανότητες του κάθε συμβόλου και επιστρέφει ένα cellarray 1x26, όπου το 26 είναι το μέγεθος του αλφαβήτου. Το κάθε cell περιέχει την κωδικοποίηση του συμβόλου.
   - **huffmanenco(inputcharacters, alphabet, encodedsymbols, length)**: δέχεται ως είσοδο την ακολουθία χαρακτήρων που πρέπει να κωδικοποιηθεί, το αλφάβητο εισόδου, το cellarray που παρήγαγε η συνάρτηση huffmandict(alphabet, probabilitymatrix) και τη μεταβλητή length που δηλώνει το μήκος του συμβόλου και επιστρέφει την κωδικοποιημένη ακολουθία.
   - **huffmandeco(encodedoutput , alphabet, encodedsymbols, length)**: δέχεται ως είσοδο την κωδικοποιημένη ακολουθία που παρήγαγε η huffmanenco(inputcharacters, alphabet, encodedsymbols, length), το αλφάβητο εισόδου, το cellarray που παρήγαγε η huffmandict(alphabet, probabilitymatrix), τη μεταβλητή length και επιστρέφει την αποκωδικοποιημένη ακολουθία χαρακτήρων που στάλθηκε.

2. **Υλοποίηση πηγής Α:**

Με την εντολή **out = randsrc(1,10000,[65:90; probabilitymatrix])** όπου ο πίνακας probabilitymatrix περιέχει τις πιθανότητες των συμβόλων της αγγλικής γλώσσας, παράγουμε 10.000 τυχαίους αριθμούς ανάμεσα στο 65 και το 90 (που είναι η κωδικοποίηση για τα γράμματα A-Z). Στη συνέχεια με την εντολή char (out) μετατρέπουμε την ακολουθία αριθμών σε γράμματα. Έτσι προκύπτει η ακολουθία εισόδου που θα κωδικοποιηθεί στα επόμενα βήματα.
Περνώντας την ακολουθία εισόδου από τις συναρτήσεις huffmanenco και huffmandeco καταλήγουμε πάλι στην αρχική ακολουθία. Αυτό δείχνει πως οι συναρτήσεις μας λειτουργούν όπως πρέπει.
Το μέσο μήκος κωδικοποιημένου συμβόλου είναι 4.2. Το συνολικό μήκος της κωδικοποιημένης ακολουθίας εισόδου είναι 42063 ψηφία (μέσος όρος από 5 εκτελέσεις).

Υλοποίηση πηγής Β:
Με την εντολή **importdata('kwords.txt')** εισάγουμε τα περιεχόμενα του αρχείου στη Matlab. Στη συνέχεια κάνουμε όλα τα γράμματα κεφαλαία και αφαιρούμε τις παύλες και τις γραμμές. Έτσι προκύπτει η ακολουθία εισόδου που θα κωδικοποιηθεί.
Περνώντας την ακολουθία εισόδου από τις συναρτήσεις huffmanenco και huffmandeco καταλήγουμε πάλι στην αρχική ακολουθία. Αυτό δείχνει πως οι συναρτήσεις μας λειτουργούν όπως πρέπει.
Το μέσο μήκος κωδικοποιημένου συμβόλου είναι 4.2. Το συνολικό μήκος της κωδικοποιημένης ακολουθίας εισόδου είναι 135481 ψηφία.

3. Σε αυτό το ερώτημα πρέπει αρχικά να υπολογίσουμε τις πιθανότητες εμφάνισης των συμβόλων βάσει του αρχείου **kwords.txt** και στη συνέχεια να γίνει το ίδιο πείραμα με το προηγούμενο ερώτημα.
   Μετράμε τον αριθμό των φορών εμφάνισης κάθε συμβόλου μέσα στο αρχείο και τον διαιρούμε με το συνολικό αριθμό γραμμάτων στο αρχείο. Με αυτόν τον πίνακα ως probabilitymatrix, εκτελούμε όπως και πριν το πείραμα.
   Προκύπτει ότι το μήκος της κωδικοποιημένης ακολουθίας είναι: 118.983.
   Το μέσο μήκος κωδικοποιημένου συμβόλου είναι 4.08.
   Παρατηρούμε ότι είναι κατα πολύ μικρότερο από το προηγούμενο ερώτημα. Αυτό ωφείλεται στις πιο αντιπροσωπευτικές πιθανότητες εμφάνισης του κάθε συμβόλου και άρα στο μικρότερο μήκος κωδικοποίησης των συχνότερων συμβόλων.
4. Η δεύτερη τάξης επέκταση πηγής είναι όταν παίρνουμε δύο σύμβολα που παράγει η πηγή και τα κωδικοποιούμε/στέλνουμε σαν ένα. Στην

περίπτωσή μας το νέο αλφάβητο περιέχει 676 σύμβολα (26x26 γιατί το αλφάβητο έχει μήκος 2).
Πάλι δημιουργώ την ακολουθία εισόδου όπως προηγουμένως με 10.000 τυχαίους χαρακτήρες, αλλά αυτή τη φορά τα κωδικοποιώ ανά δύο.
Το μέσο μήκος κωδικοποιημένου συμβόλου είναι αυτή τη φορά 8.38, και το μήκος κωδικοποιημένης ακολουθίας είναι 41.923 (μέσος όρος από 5 εκτελέσεις).
Η εντροπία της δεύτερης τάξης επέκτασης της πηγής Α είναι 8.3516.
Παρατηρούμε ότι το μέσο μήκος κωδικοποιημένου συμβόλου έχει πλησιάσει πολύ περισσότερο την εντροπία της πηγής όπως ήταν αναμενόμενο.
Επίσης, ο τύπος $H(X^n) = nH(X)$ επαληθεύεται πλήρως.

Ερώτημα 2 - Κωδικοποίηση PCM

1.

a. Στο ερώτημα αυτό έχοντας κβαντίσει την πηγή Α, μας ζητείται να υπολογίσουμε το θεωρητικό SQNR σε (db) αλλά και το πρακτικό στην έξοδο του ομοιόμορφου κβαντιστή, καθώς επίσης και να συγκρίνουμε τις τιμές αυτές. Υλοποίησα τη συνάρτηση ***sqnr.m*** η οποία εκτελεί τους παραπάνω υπολογισμούς.
   - Για το υπολογισμό του πρακτικού SQNR υπολογίσαμε το τετράγωνο της εισόδου δηλαδή το x^2 και το διαιρέσαμε με το μέγεθος του διανύσματος x δηλαδή το 10.000. Στην συνέχεια μετρήσαμε και το τετράγωνο της διαφοράς της εισόδου x με το κέντρο του διαστήματος xq στο οποίο ανήκει κάθε φορά και το διαιρέσαμε ομοίως και αυτό με το μέγεθος του διανύσματος εισόδου x δηλαδή το 10.000. Δηλαδή για τον υπολογισμό του πειραματικού SQNR εκτελέσαμε την πράξη : $E(x^2)/E[(x- x')^2]$. Το αποτέλεσμα της πράξης αυτής το μετατρέψαμε σε (db) κάνοντας χρήση του $10\log_{10}$.
   - Για τη θεωρητική τιμή του SQNR υπολογίσαμε όπως γνωρίζουμε από τη θεωρία μας **την ισχύ του σήματος** από το ολοκλήρωμα του γινομένου της συνάρτησης πυκνότητας - πιθανότητας με το τετράγωνο της εισόδου στο διάστημα που ορίζεται ο κβαντιστής. Επίσης υπολογίσαμε και **την ισχύ του θορύβου** από το άθροισμα των ολοκληρωμάτων, στα διαστήματα κβάντισης με όρια κάθε φορά τα άκρα του κάθε διαστήματος, του γινομένου της συνάρτησης πυκνότητας πιθανότητας και του τετραγώνου της διαφοράς της

εισόδου από το κεντρικό σημείο κβάντισης του διαστήματος. Στη συνέχεια με τη χρήση του $10\log_{10}$ μετατρέψαμε το αποτέλεσμα σε(db). Με λίγα πιο απλά λόγια το θεωρητικό SQNR το υπολογίσαμε από τον τύπο:

$$SQNR(db) = 10\log_{10}\frac{P_{σήματος}}{P_{θορύβου}}$$

- $P_{σήματος}$ είναι το $E[x^2] = \int x^2 f_x(x)$, με όρια ολοκλήρωσης τα όρια της δυναμικής περιοχής του κβαντιστή, στην περίπτωσή μας από [0,4].
- $P_{θορύβου}$ είναι το άθροισμα των $E[(x-\bar{x})^2] = \int (x-\bar{x})^2 f_x(x)$, με όρια κάθε φορά τα άκρα του εκάστοτε διαστήματος κβάντισης.

ΠΗΓΗ Α

N	Θεωρητικό SQNR	Πειραματικό SQNR
4 bits	*24,824*	*16,127*
6 bits	*36,962*	*17,219*

Σύμφωνα με τα παραπάνω αποτελέσματα διαπιστώνουμε μεγάλες αποκλίσεις μεταξύ της πειραματικής και της θεωρητικής τιμής του SQNR, λόγω του ότι η κβάντιση είναι βαθμωτή, σε αντίθεση με τη διανυσματική, που είναι αποδοτικότερη. Αυτό συμβαίνει λόγω του ασυμπτωτικού ορίου της συνάρτησης του ρυθμού παραμόρφωσης και ισχύει για την περίπτωση ιδανικής κβάντισης της εξόδου κατά μπλοκ, και όχι κάθε μιας εξόδου χωριστά, όπως συμβαίνει στη βαθμωτή κβάντιση.

b. Μας ζητείται να υπολογίσουμε πειραματικά την πιθανότητα υπερφόρτωσης της πηγής Α δηλαδή την πιθανότητα εκείνη στην οποία οι τιμές προς κβάντιση βρίσκονται εκτός της δυναμικής περιοχής του κβαντιστή. Για να υπολογίσουμε την παραπάνω πιθανότητα αρχικοποιήσαμε μια μεταβλητή με ονομασία *offside* με μηδέν και κάθε φορά που εντοπίζουμε μια τιμή του διανύσματος x να είναι εκτός της περιοχής του κβαντιστή η μεταβλητή offside αυξάνεται κατά 1. Στο τέλος

έχουμε καταγράψει το ποσό των τιμών εκτός ορίων και πλέον η εύρεση της πιθανότητας είναι εύκολη αφού μέσω του μαθηματικού τρόπου διαιρώντας το παραπάνω πλήθος με το πλήθος όλων των τιμών δηλαδή το μέγεθος του διανύσματος εισόδου x μας εμφανίζει την πειραματική μέτρηση. Το αποτέλεσμα της παραπάνω διαδικασίας είναι : 0,0219.

Η θεωρητική τιμή της πιθανότητας υπερφόρτωσης γεννάται λύνοντας τα παρακάτω και καταλήγοντας πως η αποδεκτή τιμή είναι το 0,0183.

$$P_{overload} = 1 - \int_0^4 e^{-x} dx = 0,0183$$

όπου: $\int_0^4 e^{-x} dx = P(0 \le X \le 4)$, η πιθανότητα να εμφανιστούν οι τιμές εντός της δυναμικής περιοχής του κβαντιστή.

2. Ο μη ομοιόμορφος κβαντιστής που κατασκευάζεται με εφαρμογή του αλγορίθμου Lloyd-Max χρησιμοποιεί ομοιόμορφη κβάντιση για την αρχικοποίηση του. Εφαρμόζουμε και την ομοιόμορφη αλλά και τη μη ομοιόμορφη κβάντιση στο σήμα φωνής και στον υπολογισμό του SQNR. Στο ερώτημα αυτό προχωρούμε στην εφαρμογή των δύο μεθόδων στο σήμα φωνής καθώς και στον υπολογισμό της τιμής του SQNR(db), δηλαδή της αναλογίας του σήματος προς το θόρυβο κβαντισμού. Για το λόγο αυτό θα πρέπει να υπολογιστεί η ισχύς του σήματος αλλά και του θορύβου. Ο πειραματικός υπολογισμός τους μπορεί να γίνει βρίσκοντας τον μέσο όρο του τετραγώνου των τιμών τους.

$$P_x = \frac{\sum_i (x[i]^2)}{\mu\acute{\eta}\kappa o\varsigma\ \sigma\acute{\eta}\mu\alpha\tau o\varsigma}, \qquad P_{noise} = D = \frac{\sum_i \left(\left(x[i] - x_q[i] \right)^2 \right)}{\mu\acute{\eta}\kappa o\varsigma\ \sigma\acute{\eta}\mu\alpha\tau o\varsigma}$$

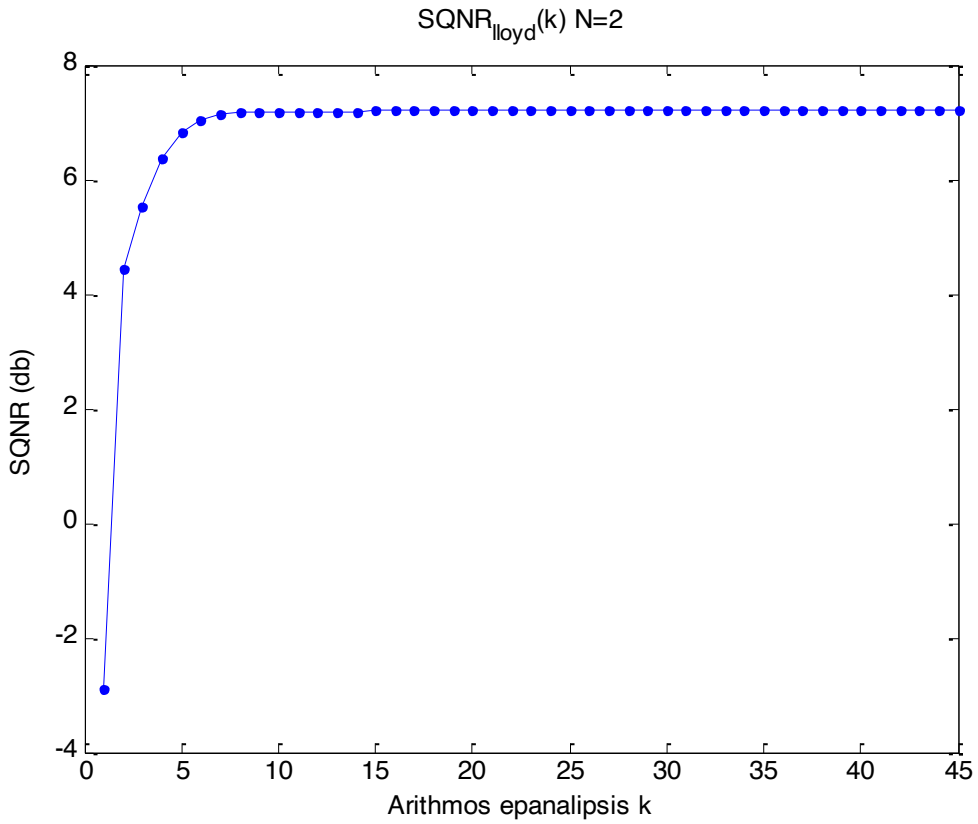
Ο λόγος SQNR(db) θα δίνεται τότε από τη σχέση:

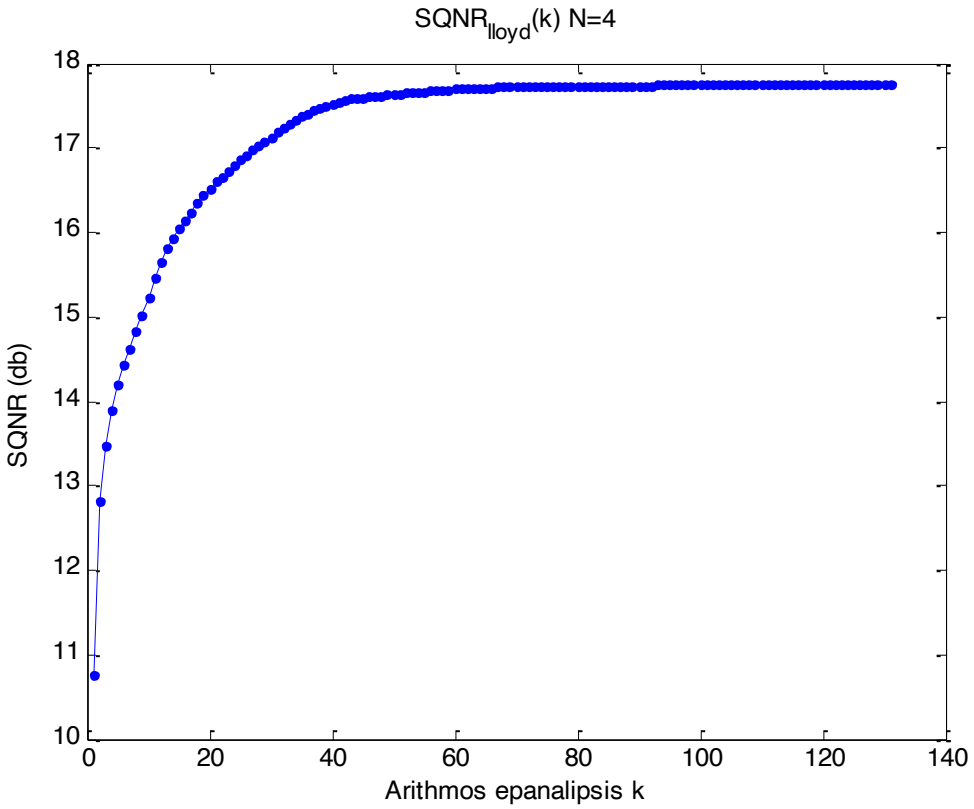
$$SQNR(db) = 10 log_{10} \left(\frac{P_x}{P_{noise}} \right)$$

Τα αποτελέσματα για την PCM κωδικοποίηση φαίνονται στον παρακάτω πίνακα:

bits	SQNR(db) Κβαντιστής PCM
2	7,195
4	17,75

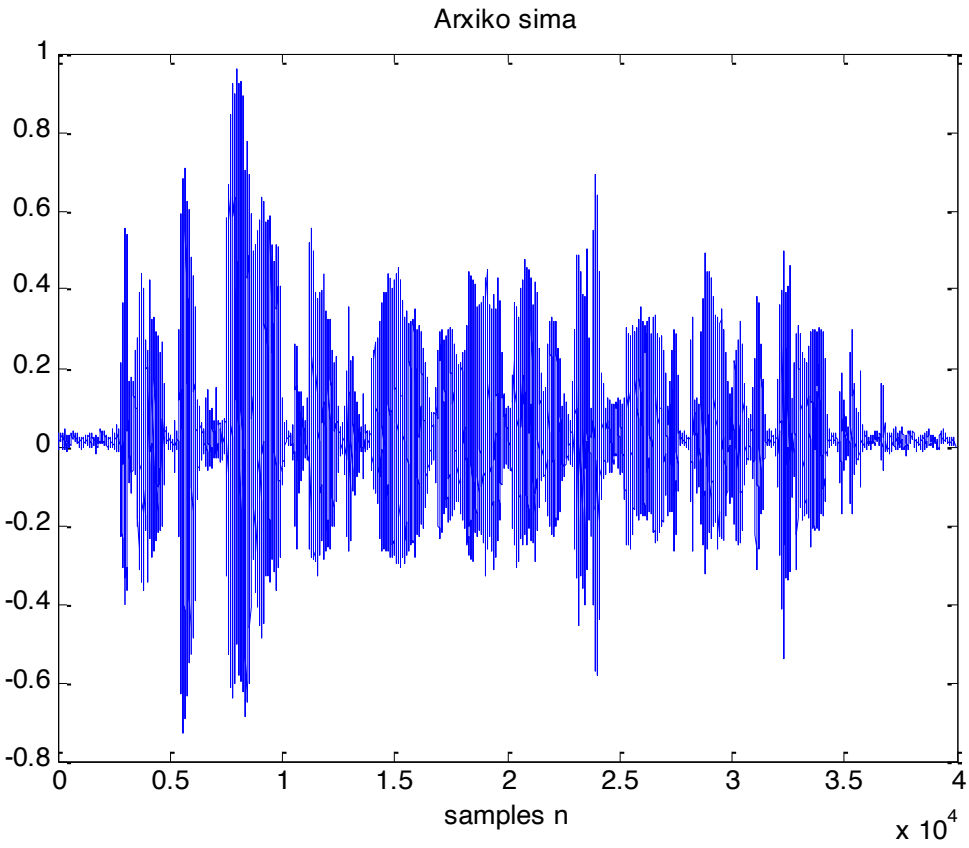
Για τον μη ομοιόμορφο κβαντιστή PCM σχεδιάζουμε τη μεταβολή του SQNR σε σχέση με τον αριθμό των επαναλήψεων:

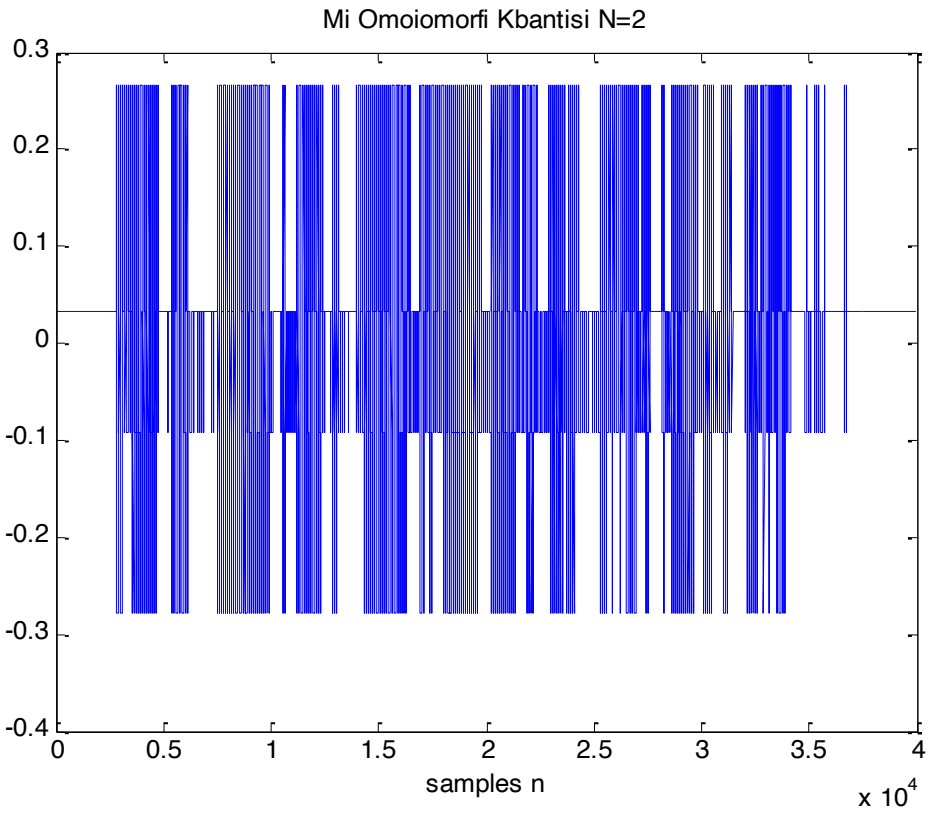


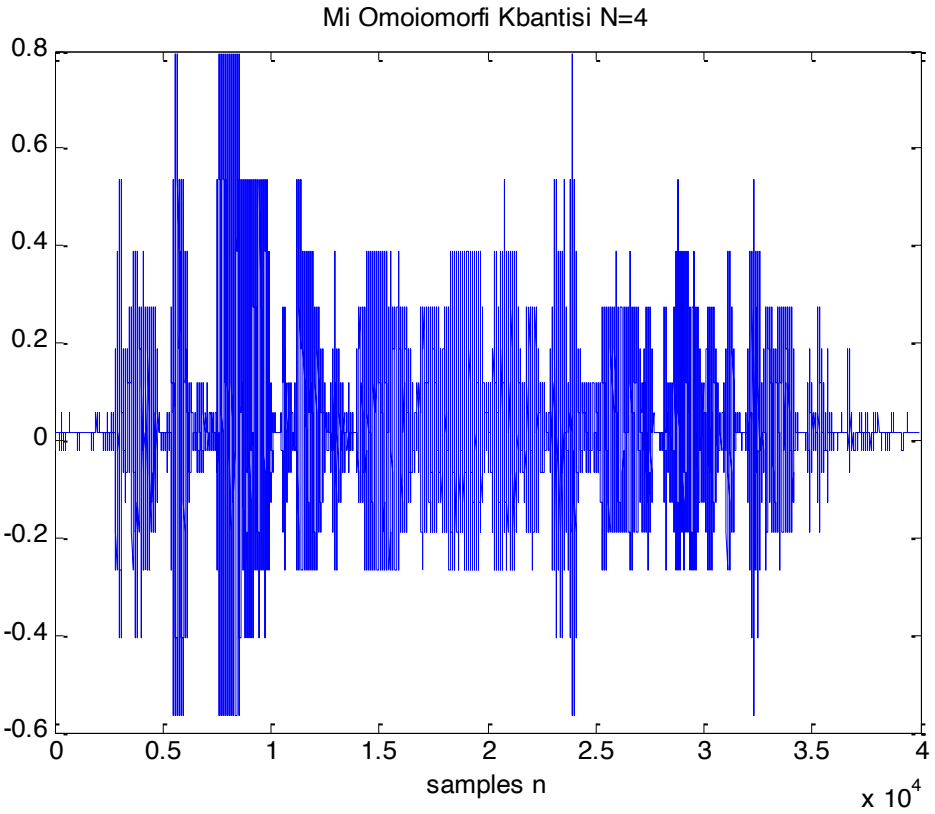


Από τις προηγούμενες γραφικές παραστάσεις μπορούμε να παρατηρήσουμε ότι όσο μεγαλώνει το πλήθος επιπέδων του κβαντιστή τόσο περισσότερο αυξάνεται το SQNR του αλλά καθυστερεί να προσεγγίσει τη μέγιστη τιμή. Αυτό είναι ένα αναμενόμενο αποτέλεσμα καθώς η αύξηση των επιπέδων του κβαντιστή μειώνει το θόρυβο του κβαντιστή.
Για N=2 το SQNR παίρνει μικρότερες τιμές σε σχέση με το SQNR για N=4.

Επίσης, για την ποιοτική αξιολόγηση των μεθόδων σχεδιάζουμε τις κυματομορφές τόσο του αρχικού ακουστικού σήματος όσο και των σημάτων που προκύπτουν από τη διαδικασία της κβάντισης:


Arxiko sima
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
samples n
x 10^4


Mi Omoiomorfi Kbantisi N=2
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
samples n
x 10^4



- Για N=2 η ομοιόμορφη κβάντιση δεν δίνει καθόλου καλό ακουστικό αποτέλεσμα αφού ο θόρυβος που δημιουργείται είναι τόσο μεγάλος που με δυσκολία μπορούμε να καταλάβουμε την πρόταση που ακούγεται. Το γεγονός αυτό επιβεβαιώνεται και από την προκύπτουσα κυματομορφή η οποία είναι σχεδόν επίπεδη και δεν θυμίζει καθόλου την αρχική.
- Για την περίπτωση N=4 βλέπουμε ότι ο ομοιόμορφος κβαντιστής προσεγγίζει γενικά τη μορφή του αρχικού σήματος και το ηχητικό αποτέλεσμα είναι ανεκτό αλλά και πάλι υπάρχει θόρυβος.

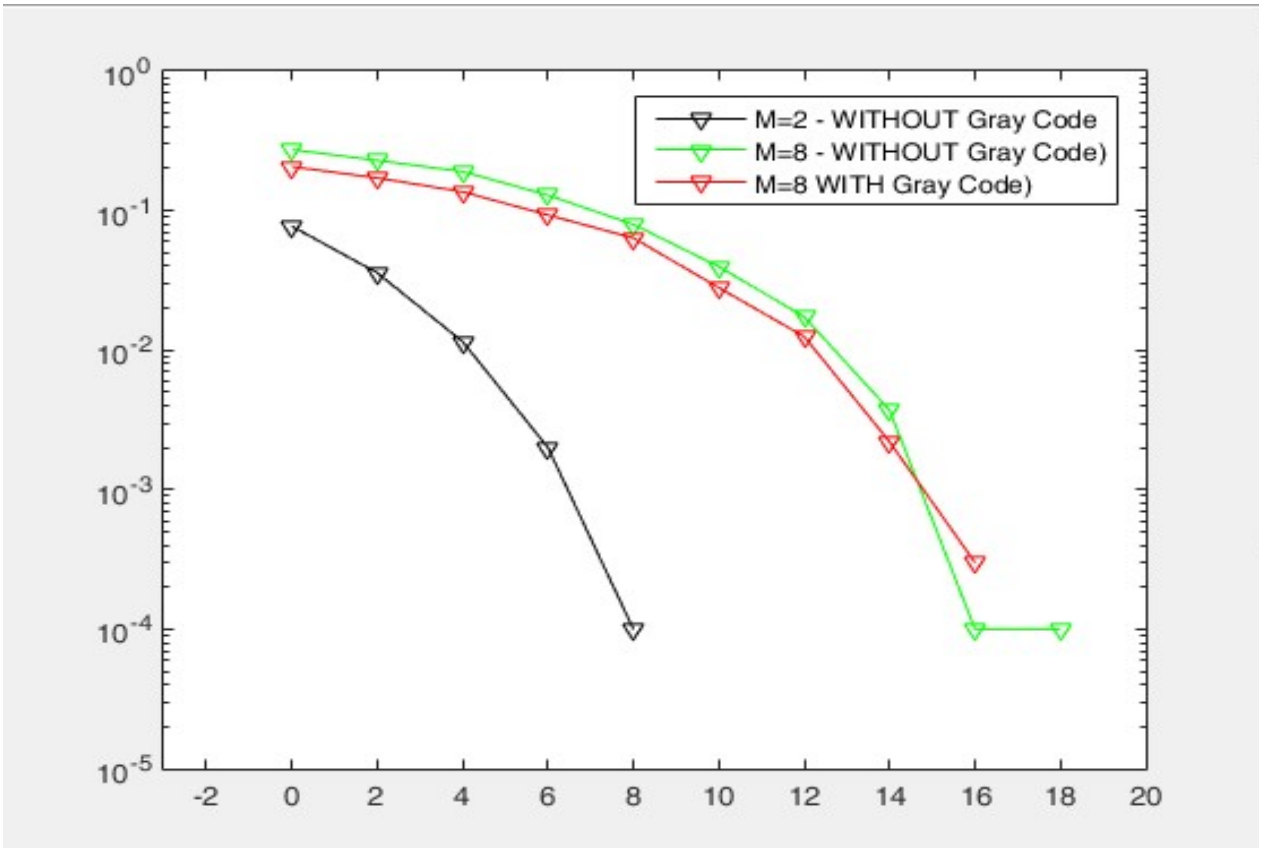
Ερώτημα 3 - Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-PAM

Σε αυτό το ερώτημα μελετήσαμε και υλοποιήσαμε ένα M-PAM σύστημα.
Το M-PAM σύστημα δέχεται ως είσοδο μια δυαδική ακολουθία την οποία τη μετατρέπει μέσω ενός κωδικοποιητή σε σύμβολα. Η κωδικοποίηση αυτή μπορεί να γίνει είτε με κλασική δυαδική σε δεκαδική κωδικοποίηση είτε με κωδικοποίηση Gray.

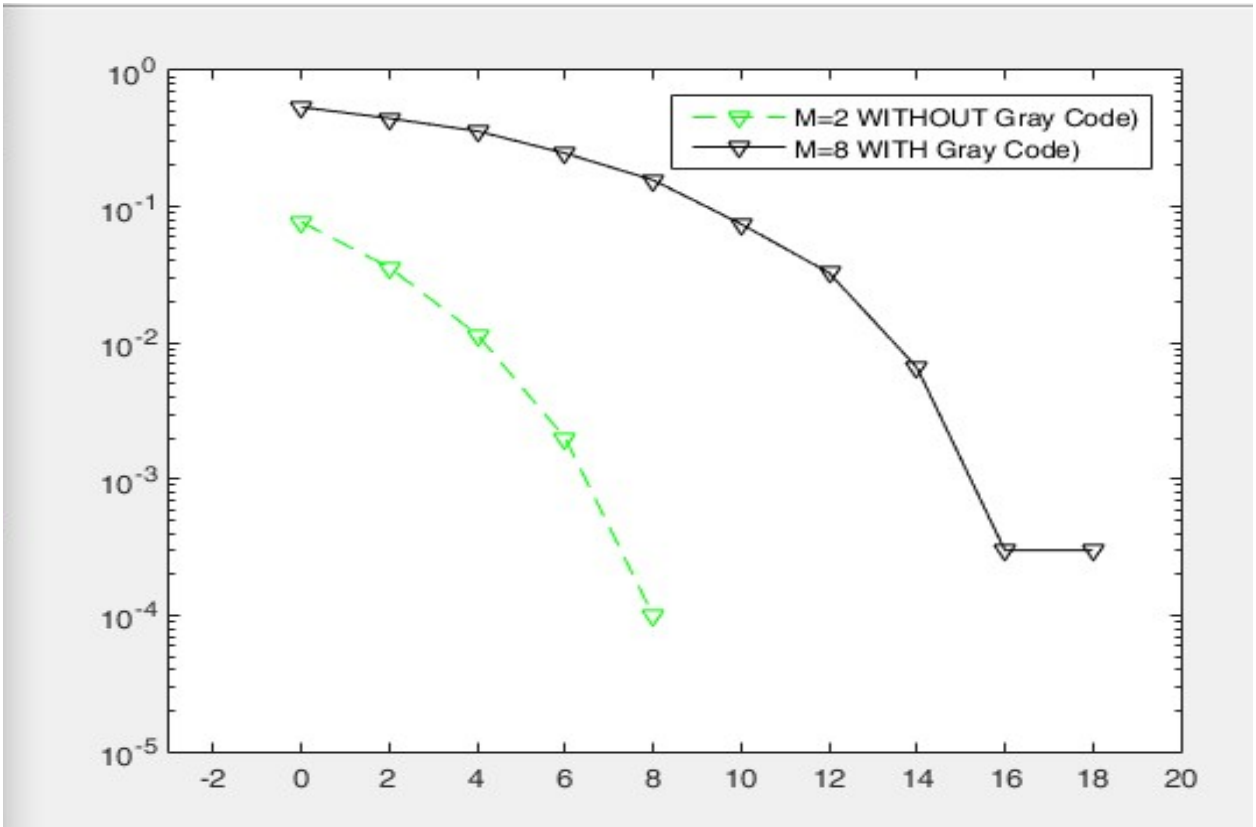
Στην κωδικοποίηση Gray, κατά τη μετάδοση του σήματος προστίθεται θόρυβος. Ο δέκτης λαμβάνει το σύμβολο συν το θόρυβο. Το ληφθέν σημείο μετατοπίζεται στη γεωμετρική αναπαράσταση του δέκτη. Η μετατόπιση γίνεται συνήθως στη γειτονιά του πραγματικού συμβόλου. Ο δέκτης μπορεί να πάρει εσφαλμένη απόφαση για το σύμβολο που στάλθηκε. Το λανθασμένο σύμβολο είναι συνήθως κάποιο από τα γειτονικά του πραγματικού συμβόλου. Συνεπώς μία εσφαλμένη απόφαση συνεπάγεται ότι ένα μόνο bit θα είναι λάθος (BER (bit error rate) < SER (symbol error rate)). Για τη διόρθωση του συμβόλου σύμφωνα με την κωδικοποίηση Gray πρέπει να αλλάξει 1 μόνο bit στο μεταδιδόμενο σύμβολο, ενώ με άλλη αντιστοίχιση πρέπει να αλλάξουν περισσότερα bit. Το πλεονέκτημα της κωδικοποίησης Gray είναι ότι ελαχιστοποιεί το BER (bit error rate).
Τα σύμβολα πολλαπλασιάζονται με ορθογώνιο παλμό και με το φέρον ημιτονοειδές σήμα cos(2πfct) και μεταδίδονται μέσω διαύλου ο οποίος προσθέτει στο μεταδιδόμενο σήμα λευκό θόρυβο που εξαρτάται από το SNR του καναλιού που προσομοιώνουμε.
Στο δέκτη τα σήματα που προκύπτουν μετά και την προσθήκη του θορύβου αποδιαμορφώνονται, δηλαδή πολλαπλασιάζονται με τον ορθογώνιο παλμό και αθροίζονται με τη φέρουσα. Από τη διαδικασία αυτή λαμβάνεται ένα σήμα που εισάγεται στο φωρατή που αποφασίζει ποιο σύμβολο απεστάλη τελικά. Τέλος, μέσω του αποκωδικοποιητή εξάγεται από τον δέκτη η ληφθείσα δυαδική ακολουθία.

Τα αποτελέσματα που πήραμε από την εκτέλεση του κώδικα για διαμόρφωση **8-PAM** είναι: **SER=0.96** και **BER=0.5.**

<u>Καμπύλη BER</u>


M=2 - WITHOUT Gray Code
M=8 - WITHOUT Gray Code)
M=8 WITH Gray Code)
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
-2
0
2
4
6
8
10
12
14
16
18
20

Καμπύλη SER


M=2 WITHOUT Gray Code)
M=8 WITH Gray Code)
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
-2
0
2
4
6
8
10
12
14
16
18
20

Γενικά, από τη θεωρία, γνωρίζουμε ότι η ελάχιστη απόσταση μεταξύ των σημείων ενός αστερισμού σημάτων επηρεάζει σημαντικά την απόδοση του αστερισμού ως προς τα σφάλματα.

- Για M=2-PAM έχουμε καλύτερη απόδοση σε σχέση με το M=8-PAM στο BER. Παρατηρούμε ότι όσο μεγαλώνει το μέγεθος του αστερισμού τόσο περισσότερο αυξάνεται το BER. Αυτό είναι αναμενόμενο γιατί όσο μικρότερη είναι η απόσταση μεταξύ των σημείων του αστερισμού τόσο μεγαλύτερη είναι η πιθανότητα σφάλματος. Όπως παρατηρούμε από τη γραφική παράσταση του BER η κωδικοποίηση Gray μειώνει ελαφρώς το BER κάτι που σημαίνει ότι μειώνει την πιθανότητα σφάλματος bit στο δέκτη.

- Για M=2-PAM έχουμε καλύτερη απόδοση σε σχέση με το M=8-PAM και στο SER. Παρατηρούμε ότι όσο μεγαλώνει το μέγεθος του αστερισμού τόσο περισσότερο αυξάνεται και το SER. Αυτό είναι επίσης αναμενόμενο γιατί όσο μεγαλύτερη είναι η απόσταση μεταξύ των συμβόλων τόσο μικρότερη είναι η πιθανότητα σφάλματος. Όμως, όπως παρατηρούμε από τη γραφική παράσταση του SER, η κωδικοποίηση Gray αυξάνει το SER κάτι που σημαίνει ότι αυξάνει την πιθανότητα σφάλματος συμβόλου στο δέκτη. Άρα η κωδικοποίηση Gray δεν έχει ευεργετικά αποτελέσματα στο SER.

Κώδικες MATLAB

Ερώτημα 1

- ✓ **huffmandict(alphabet , probabilitymatrix)**

```
function [ encodedsymbols ] = huffmandict(alphabet, probabilitymatrix)

[~ , alphabetsize] = size(alphabet); %to megethos tou alfavitou

symbolmatrix = zeros(alphabetsize, alphabetsize); %edo tha ginoun oi prostheseis ton pithanotiton ton sumbolon

symbolmatrix(1, :) = probabilitymatrix; %i proti seira einai to probabilitymatrix
encodedsymbols = repmat({''}, 1, alphabetsize);

%se kathe vima vriskoume tis 2 mikroteres pithanotites. Ti mia apo tis duo tin kanoume isi me to athroisma tous kai tin alli tin kanoume monada
for i=2:alphabetsize
[minimum,J1] = min(symbolmatrix(1,:));
symbolmatrix(1, J1) = 1;
symbolmatrix(i, J1) = -1;
[secondminimum, J2] = min(symbolmatrix(1,:));
symbolmatrix(i, J2) = 1;
symbolmatrix(1, J2) = symbolmatrix(1, J2) + minimum;
end

%ksekinontas apo to telos, paragoume tin kodikopoiisi huffman
for i=alphabetsize:-1:2
addone = find(symbolmatrix(i, :)== 1);
addzero = find(symbolmatrix(i, :)==-1);
predecessor = encodedsymbols(1, addone);
encodedsymbols(1, addzero ) = strcat(predecessor, num2str(0)); %predecessor+0
encodedsymbols(1, addone ) = strcat(predecessor, num2str(1)); %predecessor + 1
end
```

- ✓ **huffmanenco(inputcharacters, alphabet, encodedsymbols, length)**

```
function [ encodedoutput ] = huffmanenco(inputcharacters, alphabet, encodedsymbols, length)

[~, inputsize] = size(inputcharacters);

encodedoutput ='';

for i=1:slength:inputsize
pos = [];
```

```
if (length == 1)
token = inputcharacters(1, i); %lamvanoume ton i-sto xaraktira tis akolouthias eisodou

pos = find(alphabet == token);
end

if (length==2)
token = strcat(inputcharacters(1, i), inputcharacters(1, i+1));
pos2 = strfind(alphabet, token);
pos = (pos2(1, find(mod(pos2, 2) , 1))+1)/2;
end

if (isempty(pos)==0)

%synenonoume to trexon symvolo me ta proigoumena
encodedoutput = strcat(encodedoutput, encodedsymbols(1, pos));

end
end
```

- ✓ **huffmandeco(encodedoutput , alphabet, encodedsymbols, length)**

```
function [ originalcharacters ] = huffmandeco(encodedoutput , alphabet, encodedsymbols,
length)

searchfor = encodedoutput{1,1};

[~, inputsize] = size(encodedsymbols);

innercounter =1;
length(searchfor)

start=1;
end=1;

while (end<=length(searchfor))
token = searchfor(start:end);

for i=1:inputsize
if (strcmp(token, encodedsymbols{1, i})==1);

if (length == 1)
originalcharacters(1, innercounter)=alphabet(1, i);
end

if (length == 2)
originalcharacters(1, innercounter)=alphabet(1, 2*i -1);
```

```
originalcharacters(1, innercounter+1) = alphabet(1, 2*i);
innercounter=innercounter+1;
end

innercounter = innercounter +1;
start=end+1;
break;
end
end

end=end+1;
end
```

Τα **scripts** που έτρεξα για να πάρω τα αποτελέσματα δίνονται παρακάτω:

```
probabilitymatrix = [0.082 0.015 0.028 0.043 0.13 0.022 0.02 0.061 0.07 0.0015 0.0077 0.04 0.024
0.067 0.075 0.019 0.00095 0.06 0.063 0.091 0.028 0.0098 0.024 0.0015 0.02 0.00074];

alphabet = 'A':'Z';

inputcharacters = char(randsrc(1, 10000, [65:90;probabilitymatrix]));

encodedsymbols = huffmandict(alphabet, probabilitymatrix);

encodedoutput = huffmanenco(inputcharacters, alphabet, encodedsymbols, 1);

originalcharacters = huffmandeco (encodedoutput, alphabet, encodedsymbols, 1);
```

```
words=importdata('kwords.txt');
words=upper([words{:}]);
line = regexprep(words, '-','');
inputcharacters = strrep(line , '.','');

alphabet = 'A':'Z';

probabilitymatrix = [0.082 0.015 0.028 0.043 0.13 0.022 0.02 0.061 0.07 0.0015 0.0077 0.04 0.024
0.067 0.075 0.019 0.00095 0.06 0.063 0.091 0.028 0.0098 0.024 0.0015 0.02 0.00074];

encodedsymbols = huffmandict(alphabet, probabilitymatrix);

encodedoutput = huffmanenco(inputcharacters, alphabet, encodedsymbols, 1);

originalcharacters = huffmandeco(encodedoutput, alphabet, encodedsymbols, 1);
```

```
words=importdata('kwords.txt'); %ypologismos pithanotiton gia kathe gramma
words=upper( [words{:}]);
line = regexprep( words, '-','');
inputcharacters = strrep(line , '.','');

alphabet ='A':'Z';

total=sum(ismember(words, alphabet));

for k=1:numel(alphabet)
probabilitymatrix(1, k)=sum(ismember(words, alphabet(k)))/total;
end

encodedsymbols = huffmandict(alphabet, probabilitymatrix);

encodedoutput = huffmanenco(inputcharacters, alphabet, encodedsymbols, 1);

originalcharacters = huffmandeco (encodedoutput, alphabet, encodedsymbols, 1);
```

```
probabilitymatrix = [0.082 0.015 0.028 0.043 0.13 0.022 0.02 0.061 0.07 0.0015 0.0077 0.04 0.024
0.067 0.075 0.019 0.00095 0.06 0.063 0.091 0.028 0.0098 0.024 0.0015 0.02 0.00074];

alphabet = 'A':'Z';

j=1, i=1;
newalphabet = '';

for k=1:676  %dimiourgia neou alfavitou me 676 symvola
newalphabet = strcat(newalphabet, strcat(alphabet(1, j), alphabet(1, i)));
newprobabilitymatrix(1, k) = probabilitymatrix(1, j)*probabilitymatrix(1, i);

i=i+1;

if (mod(k, 26)==0)
j=j+1;
end
if (i>26)
i=1;
end
end

inputcharacters=char(randsrc(1, 10000, [65:90;probabilitymatrix]));

encodedsymbols=huffmandict(newalphabet, newprobabilitymatrix);

encodedoutput=huffmanenco(inputcharacters, newalphabet, encodedsymbols, 2);
```

```
originalcharacters=huffmandeco (encodedoutput, newalphabet, encodedsymbols, 2);
```

Ερώτημα 2.1

- ✓ **Hx.m:** συνάρτηση που υπολογίζει την εντροπία της πηγής.

```
%Συνάρτηση που μας υπολογίζει την εντροπία H(x)
%Ο τύπος της είναι για κάθε ένα σύμβολο ο υπολογισμός του αθροίσματος
% - p(i)*log2(p(i)).
function [H] = Hx(p)
H = 0; % αρχικοποιώ με μηδέν
for i=1:length(p)
   if (p(i) ~= 0)
      H = H - p(i)*log2(p(i)); % υπολογισμός εντροπίας
   end
end
```

- ✓ **source_A.m:** συνάρτηση που εκτελεί την κωδικοποίηση PCM για τις 2 διαφορετικές τιμές των bits κβάντισης της πηγής.

```
%Δημιουργώ τυχαία 10.000 δείγματα εκθετικής κατανομής σύμφωνα με την
%εκφώνηση της άσκησης. Ετσι δημιουργείται το διάνυσμα της πηγής A.
M=10000;
t = (randn(M,1)+j*randn(M,1))/sqrt(2);
x=abs(t).^2;
%Στην συνέχεια πραγματοποιούμε την ομοιόμορφη κβάντιση της πηγής A για N=4 και
6 bits
[xq_4_bits, centers_4_bits, offside(2), step(2), space_p4] = my_quantizer(x, 4, 0, 4);
[xq_6_bits, centers_6_bits, offside(3), step(3), space_p6] = my_quantizer(x, 6, 0, 4);
%-------------------------------------------------------------------%
%Ερώτημα 1-(a): Στο ερώτημα αυτό μας ζητείται να υπολογίσουμε το πειραματικο
%και το θεωρητικό sqnr εκφρασμένο σε db. Δημιουργούμε 2 διανύσματα τα
%sqnr_exper, sqnr_theor αντίστοιχα με σκοπό να καταχωρήσουμε εκει τις
%τιμές. Στην συνέχεια καλούμε την συνάρτηση sqnr που μας υπολογίζει τις
%τιμές αυτές αφού τις έχουμε ορίσει τα bits κβάντισης καθώς και την περιοχή
%στην οποία ορίζεται ο ομοιόμορφος κβαντιστής.
sqnr_exper = zeros(3,1);
sqnr_theor = zeros(3,1);
[sqnr_exper(2), sqnr_theor(2)] = sqnr(x, xq_4_bits, 4, 0, 4, centers_4_bits);
[sqnr_exper(3), sqnr_theor(3)] = sqnr(x, xq_6_bits, 6, 0, 4, centers_6_bits);

%-------------------------------------------------------------------%
%Ερώτημα 1-(b): Στο ερώτημα αυτό μας ζητήται να υπολογίσουμε την πιθανότητα
%εμφάνισης μια τιμής εκτός οριών της περιοχής του κβαντιστή. Δημιουργήσαμε
%ένα μετρητή τον οποίο ονομάσαμε offside και αυτος αυξανόταν κατά για τιμές
```

```
%μεγαλύτερες του 4 και μικρότερες του 0. Τέλος διαιρέσαμε με το 10000
%διότι το πλήθος των δειγμάτων μας είναι 10000 με σκοπό να υπολογίσουμε την
%ζητούμενη πιθανότητα.
p_uperfortosis = offside/10000;
%-------------------------------------------------------------------%

%Επίσης εκτός απο την πιθανότητα εμφάνισης κάθε στάθμης του κβαντιστή
%πρέπει να προσδιορίσουμε και την αποδοτικότητα του PCM για την πηγή A σε
%κάθε περίπτωση. Για να το πετύχουμε αυτό γνωρίζουμε απο θεωρία ότι η
%απόδοση PCM δίνεται απο τον τύπο H(x)/N οπότε το θέμα ειναι να
%υπολογίσουμε την εντροπια σε κάθε περίπτωση. Αυτο το κανουμε μέσω της
%συνάρτησης Hx.
efficiency_pcm_A = zeros(3,1);
efficiency_pcm_A4 = Hx(space_p4)/4;
efficiency_pcm_A6 = Hx(space_p6)/6;
```

- ✓ **my_quantizer.m:** συνάρτηση που υλοποιεί την κβάντιση.

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

%Ανάλογα με τον αριθμό των bits ορίζω πόσες περιοχές κβάντισης θα έχω
y = 2^N;

%Δημιουργώ το διάνυσμα xq που θα περιέχει το διάστημα που υπάρχει η κάθε
%τιμή του x ύστερα απο την κβάντιση και έχει μέγεθος 10.000 τιμές όσο
%δηλαδή και το μέγεθος του x.
xq = zeros(length(x), 1);

%Υπολογίζω το βήμα κβάντισης του κβαντιστή το οποίο δίνεται απο την διαφορά
%της μεγαλύτερης και της μικρότερης τιμής προς το πλήθος των διαστημάτων
%κβάντισης όπως το έχουμε ορίσει παρακάτω.
step = (max_value - min_value)/y;

%Μετρητής που θα μας υπολογίζει πόσες τιμές είναι εκτός της περιοχής των
%ορίων [0,4].
offside = 0;

%Υπολογισμός της πιθανότητας κάθε διαστήματος σύμφωνα με τις τιμές του x.
space_p = zeros(y, 1);

%Υπολογισμός των σημειών κάθε διαστήματος κβάντισης, τα οποία θα είναι
%σε πλήθος όσο τα διαστήματα κβάντισης + 1.
points = zeros(y+1, 1);

%Για κάθε ένα διάστημα κβάντισης θα υπάρχει και το κέντρο κβαντισής του.
centers = zeros(y, 1);
```

```
for i = 1:length(x)
%Εάν η περιοχή είναι μεγαλύτερη απο το 4 τότε αυξάνουμε την μεταβλητή
%offside κατά ένα και στην συνέχεια την ορίζουμε με 4 την μεγαλύτερη
%τιμή ώστε να μπορέσει να κβαντιστεί.
 if x(i) > max_value
  offside = offside + 1;
  x(i) = max_value;
%Εάν η περιοχή είναι μικρότερη απο το 0 τότε αυξάνουμε την μεταβλητή
%offside κατά ένα όπως και πριν και ορίζουμε με 0 την μικρότερη αυτή
%τιμή ώστε να μπορέσει να κβαντιστεί.
 elseif x(i) < min_value
       x(i) = min_value;
       offside = offside + 1;
    end
end

%Για να προσδιορίσω τις τιμές των σημείων κάθε διαστήματος θα ξεκινήσω απο
%το min_value = 0 και κάθε φορά θα προσθέτω και ένα βήμα κβάντισης.
for j = 1:y+1
   points(j) = min_value + (j-1)*step;
end

for i = 1:length(x)
   for j = 1:y

%Επειδή θα πρέπει να επιστρέψουμε και το διάνυσμα xq που περιέχει για κάθε
%μια τιμή που κβαντίζεται το διάστημα που περιέχεται, για κάθε μια τιμή
%του x ελεγχω σε ποιο διάστημα ανήκει και στην συνέχεια αυξάνω την
%πιθανότητα εμφάνισης κατά ένα στο διάστημα αυτό ώστε να μπορούμε εύκολα
%στην συνέχεια να υπολογίσουμε ακριβώς μεσω του κλασσικού μαθηματικού
%τύπου την πιθανότητα του κάθε διαστήματος.
     if (x(i)>=points(j) && x(i)<=points(j+1))
        xq(i) = j;
        space_p(j) = space_p(j) + 1;
        break;
     end
  end
end
%Υπολογισμός πιθανότητα εμφάνισης του καθε διαστήματος
space_p = space_p./length(x);

%Για να μετρήσω τις τιμές των κέντρων κβάντισης κάθε διαστήματος θα
%υπολογίσω το μέσο όρο των σημείων στα άκρα του διαστήματος που ανήκει.
for i=1:y
  centers(i) = (points(i)+ points(i+1))/2;
end
```

- ✓ **sqnr.m:** συνάρτηση που υπολογίζει τις πειραματικές και τις θεωρητικές τιμές του SQNR.

```
function [sqnr_exper, sqnr_theor] = sqnr(x, xq, N, min_value, max_value, centers)

%Σε αυτό το σημείο θα υπολογίσουμε την πειραματική τιμή του SQNR
ex = 0;
exq = 0;

for i = 1:length(xq)
   ex = ex + power(x(i),2);
   exq = exq + power(x(i)-centers(xq(i)),2);
end

%Διαιρούμε με το 10000 για να κάνουμε πειραματικό υπολογισμό
sqnr_exper = (ex/length(x)) / (exq/length(x));
sqnr_exper = 10*log10(sqnr_exper);

%Σε αυτό το σημείο θα υπολογίσουμε τη θεωρητική τιμή του SQNR δηλαδή το
%λόγο της ισχύς του σήματος προς την ισχύ του θορύβου.
n = 2^N;
step = (max_value - min_value) / n;
D = 0;
R = 0;
syms x; %ορίζω την x ώς μεταβλητή - σύμβολο
   %Υπολογισμός της ισχύς του σήματος
   F1 = (x^2)*exp(-x);
   K = int(F1, min_value, max_value);
   R = R + K;
   %Υπολογισμός της ισχύς του θορύβου
for i = 1:n
   F2 = ((x - centers(i))^2)*exp(-x);
   S = int(F2, (i-1) * step, i * step);
   D = D + S;
end

R = eval(R);
D = eval(D);

%θεωρητική τιμή του SQNR
sqnr_theor = R / D;

%Έκφραση της θεωρητικής τιμής του SQNR σε (db) σύμφωνα με την εκφώνηση της
%άσκησης.
sqnr_theor = 10*log10(sqnr_theor);
```

Ερώτημα 2.2

- **Lloyd_Max.m**

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

%x: sima eisodou
%N: arithmos bits
%max_value: megisti apodekti timi simatos eisodou
%min_value: elaxisti apodekti timi simatos eisodou
%xq: kbantismeno sima eksodou
%centers: kentra perioxwn kbantisis
%D: mesi paramorphosi stin i epanalipsi
%limits: akra perioxon kbantisis

%periorismos dynamikis perioxis simatos eisodou
x(x<min_value)=min_value;
x(x>max_value)=max_value;
M=2^N;
index=[M:-1:1];
%arxikopoiisi me xrisi omoiomorfou kbantisti
[xq,centers]=my_quantizer(x,N,min_value,max_value);
%algorithmos Lloyd_Max
kmax=200;
k=1;
while (k<kmax) && ((k<3) || (abs(D(k-1)-D(k-2))>1e-10))
   %ypologismos oriwn zwnwn kbantismou
   for i=1:M-1
      T(i)=(centers(i)+centers(i+1))/2;
   end
   %kbantisi tou simatos
   limits=flipud(T');
   [index1,xq]=quantiz(x,limits,index);
   %ypologismos mesi paramorfosis
   D(k)=mean((x-centers(xq)).^2);
   %ypologismos newn kentron kbantisis
   centers_old = centers;
   for i=1:M
      temp=find(xq==index(i));
      if ~isempty(temp)
         centers(i)=mean(x(temp));
      else
         centers(i)=centers_old(index(i));
      end
   end
   centers=flipud(centers);
   k=k+1;
end
limits = [min_value; limits; max_value];
```

- **compSQNR.m**

```
function SQNR_peir=compSQNR(x,xq,D)
%ypologismos tis peiramatikis timis tou SQNR
Px_peir=mean(x.^2);
if nargin>2
   Pnoise_peir=D;
else
   Pnoise_peir=mean((x-xq).^2);
end
SQNR_peir=10*log10(Px_peir./Pnoise_peir);
```

- **pigiB.m**

```
close all; clear; clc;
%kwdikopoisi pigis B
%fortwsi simatos
[x_A,fs,N]=wavread('speech.wav');
%kbantismos twn simatwn gia diaforetiko arithmo bits
%PCM N=2
[xqA_lloyd_2,centersA_lloyd_2,D_2]=Lloyd_Max(x_A,2,-1,1);
%PCM N=4
[xqA_lloyd_4,centersA_lloyd_4,D_4]=Lloyd_Max(x_A,4,-1,1);

%ypologismos SQNR
%N=2
SQNR_A_2_lloyd=compSQNR(x_A,[],D_2);
figure;
plot(1:length(D_2),SQNR_A_2_lloyd,'b.-');
title('SQNR_{lloyd}(k) N=2');
xlabel('Arithmos epanalipsis k');
ylabel('SQNR (db)');
%N=4
SQNR_A_4_lloyd=compSQNR(x_A,[],D_4);
figure;
plot(1:length(D_4),SQNR_A_4_lloyd,'b.-');
title('SQNR_{lloyd}(k) N=4');
xlabel('Arithmos epanalipsis k');
ylabel('SQNR (db)');

%akoustiko apotelesma kai kymatomorfes
```

```
%arxiko sima
wavplay(x_A,fs);
figure;
plot(1:length(x_A),x_A);
title('Arxiko sima');
xlabel('samples n');
pause;
%N=2
wavplay(centersA_lloyd_2(xqA_lloyd_2),fs);
figure;
plot(1:length(x_A),centersA_lloyd_2(xqA_lloyd_2));
title('Mi Omoiomorfi Kbantisi N=2');
xlabel('samples n');
pause;
%N=4
wavplay(centersA_lloyd_4(xqA_lloyd_4),fs);
figure;
plot(1:length(x_A),centersA_lloyd_4(xqA_lloyd_4));
title('Mi Omoiomorfi Kbantisi N=4');
xlabel('samples n');
pause;
```

Ερώτημα 3

Κώδικας Matlab για 2-PAM

```
%Καθορισμός χαρακτηριστικών ακολουθίας εισόδου, διαμορφωτή και Καναλιού
N = 3*10^4;
akolouthia_eisodou = randn(N,1);
M = 2;
SNR = 20;
```

```
%Metatropi ths akolouthias eisodou se bits
for i=1:N
   if (akolouthia_eisodou(i)<=0)
      akolouthia_eisodou(i)=0;
   else
      akolouthia_eisodou(i)=1;
   end
end
```

```
%Κατασκευή Κωδικοποιημένης ακολουθίας εισόδου
CODING = zeros(N,1);
for i=1:1:N
   if (akolouthia_eisodou(i)==0)
      CODING(i)=1; % symbolo 1
   elseif (akolouthia_eisodou(i)==1)
      CODING(i)=2; % symbolo 2
```

```
    end
end
%Χαρακτηριστικά Συστήματος
T_symbol = 40;
Tc = 4;
T_sample = 1;
fc=1/Tc;
Es = 1;
length_symbols=length(CODING);
symbols=CODING+1; %plithos symbolwn M

%Πλάτη συμβόλων για διαμόρφωση 2-PAM και υπολογισμός ενέργειας κάθε συμβόλου
for m=1:M
    Am(m)=(2*m-(M+1));
end
Em=sum(Am.^2)/length(Am);

%Κατασκευή ορθογώνιων παλμών
g_T=(2*repmat(symbols,1,T_symbol))-
(M+1))/sqrt(Em)*sqrt(2*Es/T_symbol).*ones(length_symbols,T_symbol);

%Κατασκευή ζωνοπερατού σήματος για μετάδοση σε ζωνοπερατό κανάλι
s_m=reshape(g_T',1,length_symbols*T_symbol).*cos(2*pi*fc*t);

%Δημιουργία AWGN θορύβου και πρόσθεση του στο σήμα εισόδου
N_0 = 1 /log2(M)/(10^(SNR/10)); awgn = sqrt(N_0 / 2) * randn(size(s_m));
s_m_plus_noise = s_m + awgn;

%Μετατροπή σήματος στο δέκτη σε διάνυσμα
for i = 1: length_symbols
    t = 1:T_symbol;
    r(i,:) = signal1(i,:) * sqrt(2/T_symbol) .* cos(2*pi*fc*t);
end

%Υπολογισμός πλάτους των ληφθέντων συμβόλων
for m=1:M
    LEX(i)= (2*m)-(M+1);
end
Em= sum(LEX.^2)/length(LEX);
LEX = LEX./sqrt(Em);
out = zeros(length_symbols,1);

%Υπολογισμός Ευκλείδειας Απόστασης ληφθέντων διανυσμάτων από τα αρχικά
διανύσματα
for i=1:length(r)
    for m=1:M
        distance(m)=norm(r(i,:)-LEX(m));
    end
[~,out(i)]=min(distance);
end
```

```
%Κατασκευή τελικής ακολουθίας στο δέκτη
for i=1:N
   %Κλασσική αποκωδικοποίηση
   if (out(i)==1)
      OUTPUT_SEQ(i)=0;
   elseif (out(i)==2)
      OUTPUT_SEQ(i)=1;
   end
end
```

```
%Υπολογισμός SER
counter = 0;
for i=1:length_symbols
    if (out(i) ~= CODING(i))
       counter = counter + 1;
    end
end
ser = counter/length_symbols;
```

```
%Υπολογισμός BER
counter = 0;
for i=1:N
   if (OUTPUT_SEQ(i) ~= akolouthia_eisodou(i))
      counter = counter + 1;
   end
end
ber = counter/length_symbols;
```

Διαφοροποίηση κώδικα Matlab για 8-PAM

```
%Κωδικοποίηση στον πομπό
CODING = zeros(N,1);
for i=1:3:N-2
   if
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3)) = 1;
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=2;
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=3;
```

```
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=4;
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=5;
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=6;
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=7;
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=8;
   end
end
```

```
%Κωδικοποίηση στο δέκτη
for p=1:N
  if (out(I)==1)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=0;
      DECODING(p*3)=0;
  elseif (out(p)==2)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=0;
      DECODING(p*3)=1;
  elseif (out(p)==3)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=1;
      DECODING(p*3)=0;
  elseif (out(p)==4)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=1;
      DECODING(p*3)=1;
  elseif (out(p)==5)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=0;
      DECODING(p*3)=0;
  elseif (out(p)==6)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=0;
      DECODING(p*3)=1;
```

```
  elseif (out(p)==7)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=1;
      DECODING(p*3)=0;
  elseif (out(p)==8)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=1;
      DECODING(p*3)=1;
  end
end
```

Κώδικας 8-PAM με κωδικοποίηση Gray

```
%Κωδικοποίηση στον πομπό
CODING = zeros(N,1);
for i=1:3:N-2
   if
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
0))
      CODING(i)=1; % symbolo 1 (000)
      CODING(fix(i/3) + mod(i,3)) = 1;
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=2; % symbolo 2 (001)
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=4; % symbolo 3 (010)
   elseif
((akolouthia_eisodou(i)==0)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=3; % symbolo 4 (011)
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=8; % symbolo 5 (100)
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==0)&&(akolouthia_eisodou(i+2)==
1))
      CODING(fix(i/3) + mod(i,3))=7; % symbolo 6 (101)
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
0))
      CODING(fix(i/3) + mod(i,3))=5; % symbolo 7 (110)
   elseif
((akolouthia_eisodou(i)==1)&&(akolouthia_eisodou(i+1)==1)&&(akolouthia_eisodou(i+2)==
1))
```

```
      CODING(fix(i/3) + mod(i,3))=6; % symbolo 8 (111)
   end
end
```

```
%Κωδικοποίηση στο δέκτη
for p=1:N
   if (out(I)==1)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=0;
      DECODING(p*3)=0;
   elseif (out(p)==2)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=0;
      DECODING(p*3)=1;
   elseif (out(p)==4)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=1;
      DECODING(p*3)=0;
   elseif (out(p)==3)
      DECODING(p*3-2)=0;
      DECODING(p*3-1)=1;
      DECODING(p*3)=1;
   elseif (out(p)==8)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=0;
      DECODING(p*3)=0;
   elseif (out(p)==7)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=0;
      DECODING(p*3)=1;
   elseif (out(p)==5)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=1;
      DECODING(p*3)=0;
   elseif (out(p)==6)
      DECODING(p*3-2)=1;
      DECODING(p*3-1)=1;
      DECODING(p*3)=1;
   end
end
```

```
%Γραφική παράσταση BER και SER
counter = 1;
myseq = 10000;

for i = 0:5:40
[pam2nogray(counter),pam2sernogray(counter)] = functions_merosb(myseq,0,2,i);
[pam8nogray(counter),pam8nograyser(counter)] = functions_merosb(myseq,0,8,i);
[pam8gray(counter),~] = functions_merosb(myseq,1,8,i);
counter = counter+1;
```

```
end

figure;
xlabel('SNR');
ylabel('BER');
semilogy([0:5:40], pam2nogray, 'k-v');
hold on;
semilogy([0:5:40], pam8nogray, 'g-v');
hold on;
semilogy([0:5:40], pam8gray, 'r-v');
axis([-3 20 10^-5 1])
legend('M=2 - WITHOUT Gray Code', 'M=8 - WITHOUT Gray Code)','M=8 WITH Gray
Code)');
hold off;

figure
xlabel('SNR');
ylabel('SER');
semilogy([0:5:40], pam2sernogray, 'g--v');
hold on;
semilogy([0:5:40], pam8nograyser, 'k-v');
axis([-3 20 10^-5 1])
legend('M=2 WITHOUT Gray Code)', 'M=8 WITH Gray Code)');
hold off;
```