Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Πανεπιστήμιο Πατρών
πολυτεχνική σχολή

Τομέας Υλικού και Αρχιτεκτονικής των Υπολογιστών
Διδάσκων: Κωνσταντίνος Μπερμπερίδης
Ακαδημαϊκό Έτος: 2024 – 2025
Ημ/νία Παράδοσης: 17/01/2025

Ψηφιακές Τηλεπικοινωνίες

1ο σετ Εργαστηριακών Ασκήσεων

Στοιχεία Φοιτητή:

Ονοματεπώνυμο:	Σωκράτης Μαντές
Α.Μ.:	1093421
E – mail:	up1093421@ac.upatras.gr
Εξάμηνο:	7

ΜΕΡΟΣ Α: Κωδικοποίηση Πηγής με τις μεθόδους PCM/DPCM

A1. 1. Υλοποιείστε το σχήμα PCM χρησιμοποιώντας (a). τον ομοιόμορφο κβαντιστή και (b). τον μη ομοιόμορφο με χρήση του αλγόριθμου Lloyd-Max.

(a). Στο συγκεκριμένο ερώτημα υλοποιούμε τον ομοιόμορφο κβαντισμό του σήματος εισόδου, το οποίο είναι ένα ψηφιακό ηχητικό σήμα υπό μορφή αρχείου κυματομορφής (.wav) το οποίο θα θεωρήσουμε ικανοποιητική αναπαράσταση του αντίστοιχου αναλογικού και περιέχεται στο αρχείο 'speech.wav'. Όπως γνωρίζουμε από την περιγραφή της εκφώνησης το αρχείο περιέχει δείγματα σήματος φωνής με ρυθμό δειγματοληψίας fs = 8 KHz κβαντισμένα με N = 16 bits (PCM κωδικοποίηση).Ο κβαντισμός είναι μια διαδικασία που μειώνει την ακρίβεια ενός αναλογικού σήματος για να μπορεί να αποθηκευτεί ή να μεταδοθεί ψηφιακά.

Για τον συγκεκριμένο σκοπό δημιουργήθηκε το αρχείο PCM_quantizer.m όπου θα υλοποιείται η διαδικασία ομοιόμορφου κβαντισμού. Αρχικά, ορίζουμε τη συνάρτηση my_quantizer, η οποία δέχεται τα εξής ορίσματα:

x: το σήμα εισόδου

N: ο αριθμός των bits για τον κβαντισμό

min_value,max_value: τα όρια των αποδεκτών τιμών του σήματος.

Ως εξόδους δίνει τους δείκτες των επιπέδων κβαντισμού για κάθε δείγμα του σήματος(xq) και τα κέντρα των επιπέδων κβαντισμού(centers).

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

Έπειτα, όλα τα δείγματα του σήματος περιορίζονται στα όρια {min_value},{max_value}. Αυτό αποτρέπει την υπερχείλιση. Ο συνολικός αριθμός των επιπέδων κβαντισμού (L) υπολογίζεται ως L = 2^N, όπου N είναι τα bits. Στη συνέχεια, ορίζουμε το βήμα κβαντισμού (Δ), που είναι η απόσταση μεταξύ δύο διαδοχικών επιπέδων και καθορίζει την ακρίβεια του κβαντισμού. Και τέλος υλοποιούμε μια ομοιόμορφη κατανομή των L επιπέδων κβαντισμού εντός του εύρους {min_value},{max_value} για να δημιουργήσουμε τα κέντρα των περιοχών κβαντισμού. Τα κέντρα περιοχών κβαντισμού είναι οι αντιπροσωπευτικές τιμές για κάθε επίπεδο κβαντισμού. Στον ομοιόμορφο κβαντισμό, κατανέμονται συμμετρικά και με ίσες αποστάσεις μεταξύ τους (βήμα Δ). Το πρώτο κέντρο θα βρίσκεται στη μέση της πρώτης περιοχής κβαντισμού και το τελευταίο κέντρο βρίσκεται στη μέση της τελευταίας περιοχής. Η τοποθέτηση στη μέση κάθε περιοχής εξασφαλίζει ότι ελαχιστοποιείται το Σφάλμα Ανακατασκευής, δηλαδή όταν ένα δείγμα σήματος ανακατασκευάζεται, χρησιμοποιείται η τιμή του κέντρου. Εάν το κέντρο είναι στη μέση της περιοχής, το σφάλμα ανακατασκευής (διαφορά μεταξύ της πραγματικής και της ανακατασκευασμένης τιμής) ελαχιστοποιείται. Αυτή η διαδικασία ελαχιστοποιεί το σφάλμα |x−center|, με αποτέλεσμα την καλύτερη ποιότητα σήματος για δεδομένο αριθμό bits.

```
% Περιορισμός του σήματος στις αποδεκτές τιμές [min_value, max_value]
    x = max(min(x, max_value), min_value);

    % Υπολογισμός του αριθμού επιπέδων κβαντισμού
    L = 2^N;

    % Υπολογισμός του βήματος κβαντισμού
    delta = (max_value - min_value) / L;

    % Υπολογισμός των κέντρων των περιοχών κβαντισμού
    centers = linspace(min_value + delta / 2, max_value - delta / 2, L);
```

Υπολογίζεται το διάνυσμα εξόδου xq που αντιστοιχεί στο επίπεδο του κάθε δείγματος και διορθώνονται οι περιπτώσεις όπου xq=0 (τιμές στο ελάχιστο όριο) ή xq> (τιμές στο μέγιστο όριο). Θα ολοκληρώσουμε την υλοποίηση της συνάρτησης επιστρέφοντας το διάνυσμα εξόδου και των κέντρων κβαντισμού.

```
% Κατανομή κάθε δείγματος στην κατάλληλη περιοχή κβαντισμού
    xq = ceil((x - min_value) / delta);
    xq(xq == 0) = 1;  % Για τιμές ακριβώς στο min_value
    xq(xq > L) = L;   % Για τιμές ακριβώς στο max_value

    % Επιστροφή του διανύσματος εξόδου xq και των κέντρων κβαντισμού
end
```

Για να εκτελέσουμε τον ομοιόμορφο κβαντιστή δημιουργούμε και ένα αρχείο run_quantizer.m στο οποίο θα φορτώνεται το αρχείο speech.wav. Ορίζουμε την μεταβλητή y που περιέχει μια σειρά από δείγματα ήχου και την μεταβλητή fs που είναι η συχνότητα δειγματοληψίας. Ορίζουμε τις παραμέτρους του κβαντιστή και κανονικοποιούμε το σήμα ώστε όλες οι τιμές να βρίσκονται στο διάστημα [−1,1] αλλιώς θα υπήρχε απώλεια πληροφορίας κατά τον κβαντισμό. Ακόμα με την κανονικοποιήση η σχετική αναλογία μεταξύ των τιμών του διατηρείται, αυτό σημαίνει ότι η «μορφή» του σήματος δεν αλλάζει. Οπότε, ορίζουμε τις δυο συναρτήσεις abs(y), max(), όπου η συνάρτηση abs(y) επιστρέφει τις απόλυτες τιμές όλων των δειγμάτων του σήματος και η συνάρτηση max() βρίσκει τη μέγιστη τιμή από αυτές. Μετά κάθε δείγμα y[i] διαιρείται με τη μέγιστη απόλυτη τιμή σύμφωνα με τον τύπο:

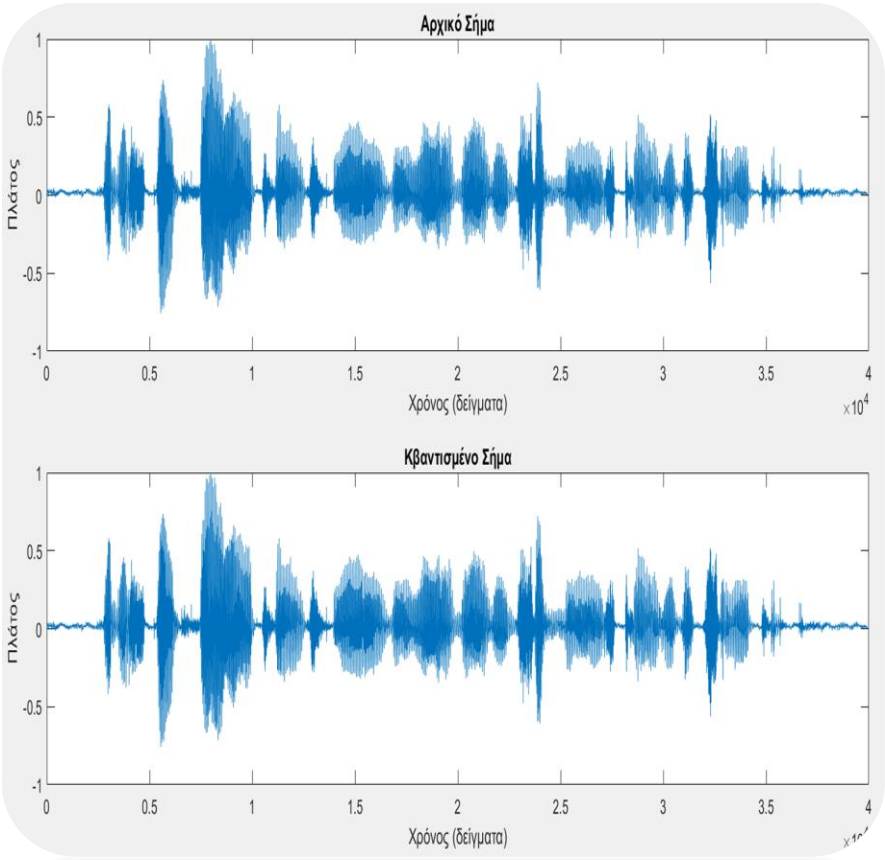
$$y_{norm}[i] = \frac{y[i]}{\max(|y|)}$$

```
% Φόρτωση του σήματος από το αρχείο
[y, fs] = audioread('speech.wav');

% Ορισμός παραμέτρων του κβαντιστή
min_value = -1;                    % Ελάχιστη τιμή του σήματος
max_value = 1;                     % Μέγιστη τιμή του σήματος
N = 4;                             % Αριθμός bits

% Κανονικοποίηση του σήματος
y = y / max(abs(y));
```

Εφαρμόζουμε τη διαδικασία κβαντισμού στο σήμα y με τη συνάρτηση my_quantizer και ανακατασκευάζουμε το σήμα από τους δείκτες κβαντισμού. Κάθε δείκτης xq αντιστοιχεί στην τιμή του κέντρου της περιοχής κβαντισμού. Τελικά, καλούμε να εκτυπωθούν τα αποτελέσματα του κβαντισμού, δηλαδή ο αριθμός των bits (N), ο αριθμός επιπέδων κβαντισμού και το βήμα κβαντισμού (Δ).

```
% Εκτέλεση του ομοιόμορφου κβαντιστή
[xq, centers] = my_quantizer(y, N, min_value, max_value);

% Ανακατασκευή του κβαντισμένου σήματος
reconstructed_signal = centers(xq);

% Εμφάνιση αποτελεσμάτων κβαντισμού
fprintf('Αποτελέσματα Κβαντισμού:\n');
fprintf('Αριθμός Bits: %d\n', N);
fprintf('Επίπεδα Κβαντισμού: %d\n', 2^N);
fprintf('Βήμα Κβαντισμού (Delta): %.4f\n', (max_value - min_value) / (2^N));
```

Τρέχοντας το αρχείο run_quantizer.m για τα πρώτα 10 δείγματα του αρχικού σήματος λαμβάνουμε τα εξής αποτελέσματα:




```
>> run_quantizer
Αποτελέσματα Κβαντισμού:
Αριθμός Bits: 4
Επίπεδα Κβαντισμού: 16
Βήμα Κβαντισμού (Delta): 0.1250
Πρώτα 10 δείγματα του αρχικού σήματος:
         0
         0
   -0.0000
    0.0001
   -0.0001
   -0.0005
    0.0150
    0.0162
    0.0186
    0.0112

Πρώτα 10 δείκτες κβαντισμού:
     8
     8
     8
     9
     8
     8
     9
     9
     9
     9

Πρώτα 10 τιμές ανακατασκευασμένου
σήματος:
   -0.0625   -0.0625   -0.0625    0.0625
-0.0625   -0.0625    0.0625    0.0625
0.0625    0.0625

Ακούγεται το αρχικό σήμα...
Ακούγεται το κβαντισμένο σήμα...
```

ΠΑΡΑΤΗΡΗΣΗ:

Το πάνω διάγραμμα (Αρχικό Σήμα) αναπαριστά το κανονικοποιημένο σήμα πριν τον κβαντισμό. Περιέχει όλες τις αρχικές πληροφορίες του ήχου με πλήρη ακρίβεια.

Το κάτω διάγραμμα (Κβαντισμένο Σήμα) αναπαριστά το σήμα μετά τον κβαντισμό. Το σήμα είναι πιο "σκαλωτό" καθώς οι τιμές έχουν προσεγγιστεί από τα επίπεδα κβαντισμού. Αυτή η προσέγγιση αντικατοπτρίζει τη διαδικασία απώλειας ακρίβειας κατά τον κβαντισμό.

Οι δύο καμπύλες έχουν παρόμοια δομή και σχήμα, κάτι που μας δείχνει ότι η διαδικασία κβαντισμού λειτουργεί σωστά. Γενικά, μικρές αποκλίσεις μεταξύ των δύο καμπυλών είναι αναμενόμενες, καθώς ο κβαντισμός προσδιορίζει συγκεκριμένα επίπεδα για τις τιμές του σήματος.

(b). Αυτή τη φορά υλοποιούμε ένα μη ομοιόμορφο κβαντισμό του διανύσματος εισόδου χρησιμοποιώντας τον αλγόριθμο Lloyd-Max, ο οποίος επιτρέπει την σχεδίαση βέλτιστου κβαντιστή για οποιοδήποτε αριθμό επιπέδων. Τα επίπεδα και τα όρια κβαντισμού προσαρμόζονται στη στατιστική κατανομή του σήματος και η διαδικασία είναι επαναληπτική. Για τον συγκεκριμένο σκοπό δημιουργήθηκε το αρχείο Lloyd_max_quantizer.m όπου θα υλοποιείται η διαδικασία μη ομοιόμορφου κβαντισμού. Ξεκινάμε ορίζοντας την συνάρτηση ορίζουμε τη συνάρτηση my_lloyd_max_quantizer, η οποία δέχεται σαν ορίσματα τα ίδια με την my_quantizer προσθέτοντας επιπλέον το όρισμα max_iter που είναι ο μέγιστος αριθμός επαναλήψεων για τη σύγκλιση.

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

Περιορίζουμε ξανά το σήμα στα επιθυμητά όρια και στην συνέχεια υπολογίζουμε ξανά τα αρχικά επίπεδα κβαντισμού. Εδώ τα L κέντρα κβαντισμού κατανέμονται ισομερώς στο διάστημα και η χρήση της συνάρτησης linspace δημιουργεί μια γραμμική κατανομή των κέντρων. Για να ορίσουμε τα όρια στην κβάντηση δημιουργούμε τον πίνακα thresholds στον οποίο θα αποθηκεύονται τα όρια των περιοχών κβαντισμού και αρχικοποιείται με μηδενικά, ώστε να μπορούν να οριστούν τα όρια στις επόμενες επαναλήψεις του αλγορίθμου.Το μήκος του πίνακα είναι L+1, όπου L=2^N είναι ο αριθμός των επιπέδων κβαντισμού. Το πρώτο όριο thresholds(1) ορίζεται ως $-\infty$ για να περιλαμβάνει όλες τις τιμές μικρότερες από το πρώτο κέντρο.Το τελευταίο όριο thresholds(L+1) ορίζεται ως $+\infty$ για να περιλαμβάνει όλες τις τιμές μεγαλύτερες από το τελευταίο κέντρο.

```
% Περιορισμός του σήματος στις αποδεκτές τιμές [min_value, max_value]
    x = max(min(x, max_value), min_value);

    % Υπολογισμός του αριθμού επιπέδων κβαντισμού
    L = 2^N;

    % Αρχικοποίηση των κέντρων (ισομερώς κατανεμημένα)
    centers = linspace(min_value, max_value, L);

    % Αρχικοποίηση των ορίων
    thresholds = zeros(1, L + 1);
    thresholds(1) = -Inf;
    thresholds(end) = Inf;
```

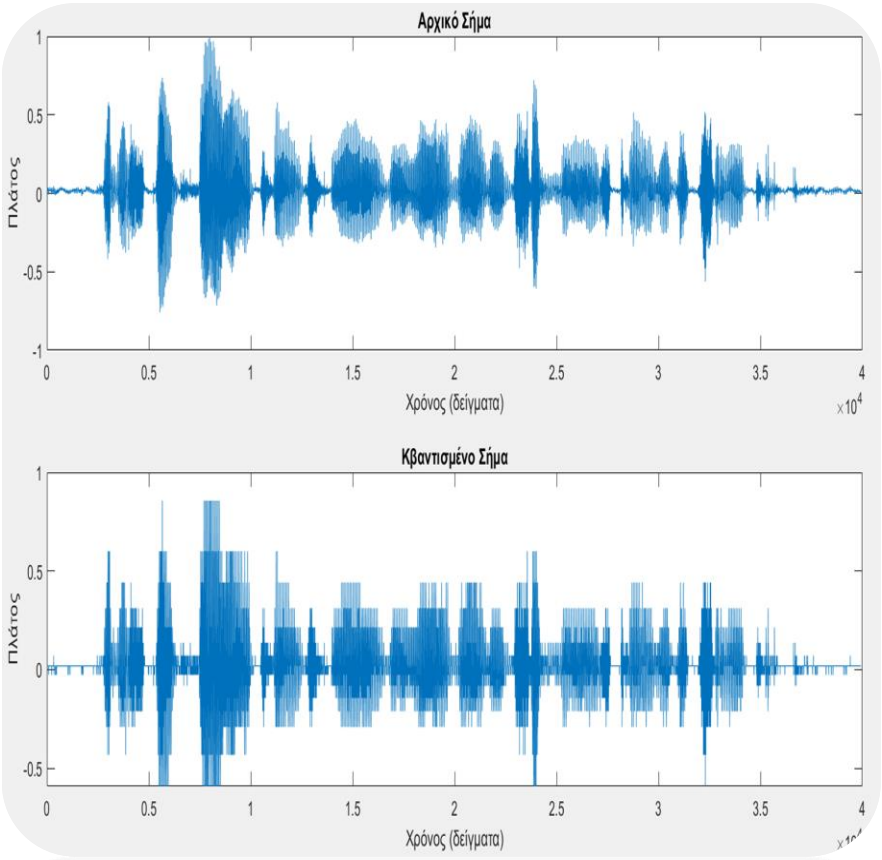
Προχωράμε στην εφαρμογή του επαναληπτικού αλγορίθμου. Ορίζουμε έναν μετρητή iter για τον αριθμό των επαναλήψεων όπου max_iter ο μέγιστος αριθμός επαναλήψεων που επιτρέπει ο αλγόριθμος (για να αποφευχθεί η περίπτωση της ατέρμονης επανάληψης). Εφαρμόζουμε ένα for loop με τις επαναλήψεις να παίρνουν τιμές από το 2 μέχρι το L (η τιμή 1 παραλείπεται γιατί χρησιμοποιείται συχνά για την αρχική θέση που δεν χρειάζεται επεξεργασία) μέσα στο οποίο τα όρια thresholds(i) υπολογίζονται ως το μέσο σημείο μεταξύ δύο διαδοχικών κέντρων centers(i−1) και centers(i). Αυτό διασφαλίζει ότι κάθε τιμή x που βρίσκεται σε μία περιοχή [thresholds(i),thresholds(i+1)] αντιστοιχίζεται σωστά στο πλησιέστερο κέντρο. Για κάθε επίπεδο i: Εντοπίζουμε τις τιμές του x που ανήκουν στην περιοχή [thresholds(i),thresholds(i+1)]. Αναθέτουμε στο xq τον δείκτη του αντίστοιχου επιπέδου i και ο πίνακας xq περιέχει πλέον τους δείκτες κβαντισμού για κάθε δείγμα του σήματος x. Στη συνέχεια, δημιουργούμε έναν πίνακα new_centers για την αποθήκευση των νέων τιμών των κέντρων, ο οποίος αρχικοποιείται επίσης με μηδενικά για να εξασφαλιστεί ότι έχει σωστό μέγεθος πριν γεμίσει με δεδομένα. Για κάθε επίπεδο i, εντοπίζουμε τις τιμές του x που ανήκουν στην περιοχή [thresholds(i),thresholds(i+1)]. Υπολογίζουμε το νέο κέντρο new_centers(i) ως τον μέσο όρο των τιμών αυτών αν η περιοχή δεν είναι κενή. Με αυτό τον τρόπο το νέο κέντρο τοποθετείται "κεντρικά" στις τιμές της περιοχής, εξασφαλίζοντας ότι το σφάλμα ανακατασκευής μειώνεται. Αντίστοιχα, αν η περιοχή είναι κενή το νέο κέντρο new_centers(i) παραμένει ίσο με την προηγούμενη τιμή centers(i). Έτσι, αποφεύγεται η ανεπιθύμητη μετατόπιση του κέντρου σε περιπτώσεις όπου δεν υπάρχουν τιμές στην περιοχή. Μετά την ολοκλήρωση του βρόχου, ο πίνακας new_centers περιέχει τις νέες τιμές για τα κέντρα.

```
% Lloyd-Max επαναληπτικός αλγόριθμος
    for iter = 1:max_iter
        % Υπολογισμός των ορίων βάσει των κέντρων
        for i = 2:L
            thresholds(i) = (centers(i-1) + centers(i)) / 2;
        end

        % Αντιστοίχιση των τιμών του σήματος σε περιοχές
        xq = zeros(size(x));
        for i = 1:L
            indices = x > thresholds(i) & x <= thresholds(i+1);
            xq(indices) = i;
        end

        % Αναπροσαρμογή των κέντρων
        new_centers = zeros(1, L);
        for i = 1:L
            region = x(x > thresholds(i) & x <= thresholds(i+1));
            if ~isempty(region)
                new_centers(i) = mean(region);
            else
                new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
            end
        end
```

Έχουμε ως σκοπό όμως την εξοικονόμηση υπολογιστικών πόρων, οπότε πρέπει να σταματάει η επανάληψη όταν τα κέντρα έχουν σταθεροποιηθεί. Για αυτό τον λόγο υπολογίζουμε τη μέγιστη διαφορά μεταξύ των παλιών κέντρων centers και των νέων new_centers. Αν η μέγιστη διαφορά είναι μικρότερη από 10^(-6) θεωρείται ότι ο αλγόριθμος έχει συγκλίνει και σταματά η επανάληψη. Τέλος, ενημερώνουμε τα παλιά κέντρα centers με τις νέες τιμές new_centers, ώστε να χρησιμοποιηθούν στην επόμενη επανάληψη.

```
% Έλεγχος για σύγκλιση
        if max(abs(new_centers - centers)) < 1e-6
            break;
        end

        centers = new_centers;
    end

    % Επιστροφή του διανύσματος εξόδου xq και των κέντρων κβαντισμού
end
```

Για να εκτελέσουμε τον μη ομοιόμορφο κβαντιστή δημιουργούμε και ένα αρχείο run_lloyd_max_quantizer.m στο οποίο θα φορτώνεται ξανά το αρχείο speech.wav. Το αρχείο αυτό υλοποιείται ακριβώς με τον ίδιο τρόπο όπως και για το run_quantizer.m.
Οπότε, τρέχοντας το αρχείο run_lloyd_max_quantizer.m για μέγιστο αριθμό επαναλήψεων max_iter ίσο με 50 λαμβάνουμε τα εξής αποτελέσματα:




```
>> run_loyd_max_quantizer
Αποτελέσματα Κβαντισμού (Lloyd-Max):
Αριθμός Bits: 4
Επίπεδα Κβαντισμού: 16
Πρώτα 10 κέντρα κβαντισμού:
   -1.0000   -0.5916   -0.4316   -0.2909
-0.2112   -0.1437   -0.0814   -0.0261
0.0185    0.0711

Πρώτα 10 δείγματα του αρχικού σήματος:
         0
         0
   -0.0000
    0.0001
   -0.0001
   -0.0005
    0.0150
    0.0162
    0.0186
    0.0112

Πρώτα 10 δείκτες κβαντισμού:
     9
     9
     9
     9
     9
     9
     9
     9
     9
     9

Πρώτα 10 τιμές ανακατασκευασμένου
σήματος:
    0.0185    0.0185    0.0185    0.0185
0.0185    0.0185    0.0185    0.0185
0.0185    0.0185

Ακούγεται το αρχικό σήμα...
Ακούγεται το κβαντισμένο σήμα...
```

ΠΑΡΑΤΗΡΗΣΗ:

Το πάνω διάγραμμα (Αρχικό Σήμα) αναπαριστά το αρχικό σήμα μετά την κανονικοποίηση. Περιέχει την πλήρη ακρίβεια του σήματος, χωρίς καμία απώλεια πληροφορίας.

Το κάτω διάγραμμα (Κβαντισμένο Σήμα) αναπαριστά το σήμα που έχει περάσει από τον μη ομοιόμορφο κβαντισμό Lloyd-Max. Οι τιμές του σήματος περιορίζονται σε συγκεκριμένα επίπεδα κβαντισμού (τα κέντρα).

Το αποτέλεσμα είναι "σκαλωτό", καθώς κάθε τιμή x[i] του σήματος αντιστοιχίζεται στο πλησιέστερο κέντρο.

Το κβαντισμένο σήμα έχει παρόμοιο σχήμα με το αρχικό, αλλά με λιγότερα επίπεδα ακρίβειας ενώ σε περιοχές του σήματος με περισσότερες τιμές, τα επίπεδα κβαντισμού θα είναι πιο "πυκνά".

A1. 2. Κωδικοποιείστε την πηγή ήχου για min_value = -1, max_value = 1 και N = 2, 4 και 8 bits. Αξιολογείστε το σχήμα PCM βασισμένοι στα ακόλουθα:
i.Σχεδιάστε για το σχήμα κβάντισης (b) πώς μεταβάλλεται το SQNR σε σχέση με τον αριθμό των επαναλήψεων του αλγορίθμου Kmax (χρησιμοποιήστε οποιαδήποτε τιμή ε = [10-16, 10-6]). Απεικονίστε στο ίδιο γράφημα τις καμπύλες μεταβολής του SQNR για κάθε διαφορετική τιμή N ώστε να μπορεί να γίνει η σύγκριση χρησιμοποιώντας διαφορετικά χρώματα, τύπο γραμμής και σχετικές λεζάντες που προσφέρει η συνάρτηση plot() στο Matlab.
ii. Για το σχήμα κβάντισης (b) συγκρίνετε τη τιμή του SQNR μετά από Kmax επαναλήψεις με αυτή που προκύπτει αν χρησιμοποιήσουμε το σχήμα (a) (ομοιόμορφη κβάντιση). Υπολογίστε και σχολιάστε την απόδοση των δύο κβαντιστών.
iii. Αξιολογείστε το αποτέλεσμα για κάθε κβαντιστή χρησιμοποιώντας την sound() (ακουστικό αποτέλεσμα) και τις κυματομορφές εξόδου. Η σύγκριση να γίνει σε σχέση με το αρχικό σήμα (χωρίς κωδικοποιήση) και για κάθε τιμή του N. Για κάθε N, οι κυματομορφές του αρχικού και του κωδικοποιημένου σήματος να απεικονιστούν στο ίδιο γράφημα με διαφορετικά χρώματα και τύπο γραμμής ώστε να μπορεί να γίνει η σύγκριση.
iv. Σχολιάστε σε κάθε τύπο κβαντιστή την αποδοτικότητα της κωδικοποίησης PCM βασισμένοι στο Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error) σε σχέση με τις τιμές του N. Απεικονίστε στο ίδιο γράφημα τις καμπύλες MSEγια τους δύο κβαντιστές με διαφορετικά χρώματα και τύπο γραμμής ώστε να μπορεί να γίνει η σύγκριση.
Το τελικό SQNR που υπολογίζετε να δίνεται σε dB, δηλαδή στη μορφή 10log10(.).

(i). Οι βασικοί στόχοι της υλοποίησης είναι ο βέλτιστος κβαντισμός με εφαρμογή του Lloyd-Max για τη βέλτιστη τοποθέτηση επιπέδων κβαντισμού με ελαχιστοποίηση του σφάλματος και η ανάλυση της αποδοτικότητας εξετάζοντας τη σχέση του SQNR με τον αριθμό bits (N) και τις επαναλήψεις (Kmax). Αρχικά, ο κώδικας ξεκινά με τη φόρτωση του σήματος speech.wav το οποίο κανονικοποιείται στο διάστημα [−1,1] για να διευκολυνθεί η ομοιόμορφη αρχικοποίηση των κέντρων και των ορίων. Αυτή η κανονικοποίηση είναι απαραίτητη για τη σωστή λειτουργία του Lloyd-Max. Οι παράμετροι που ορίζονται περιλαμβάνουν ξανά τις παραμέτρους min_value και max_value (τα όρια της δυναμικής περιοχής του σήματος ([−1,1])), N_bits (ο αριθμός bits του κβαντιστή (N=2,4,8)) και max_iters (ο αριθμός επαναλήψεων του Lloyd-Max (Kmax =10,50,100,200)). Κάθε συνδυασμός N και Kmax αναλύεται ξεχωριστά.

```
% Φόρτωση του αρχείου ήχου
[y, fs] = audioread('speech.wav');
y = y / max(abs(y));               % Κανονικοποίηση του σήματος στο [-1, 1]

% Παράμετροι
min_value = -1;                    % Ελάχιστη τιμή
max_value = 1;                     % Μέγιστη τιμή
N_bits = [2, 4, 8];                % Bits κβαντισμού
max_iters = [10, 50, 100, 200];    % Αριθμός επαναλήψεων
```

Προχωράμε στον υπολογισμό και τη σχεδίαση του SQNR για διαφορετικά N. Δημιουργούμε έναν πίνακα SQNR_results διαστάσεων length(N_bits)×length(max_iters),ο οποίος θα αποθηκεύσει τις τιμές του SQNR για κάθε συνδυασμό N (bits κβαντισμού) και Kmax (επανάληψη Lloyd-Max). Αυτό χρειάζεται επειδή για κάθε N και Kmax υπολογίζεται μία μοναδική τιμή SQNR, οπότε απαιτείται ένας πίνακας για την αποθήκευση όλων των αποτελεσμάτων που θα χρησιμοποιηθούν για τη σχεδίαση. Ο εξωτερικός βρόχος για τις επαναλήψεις Kmax διατρέχει όλες τις τιμές του Kmax που καθορίζονται από τον πίνακα max_iters και επιτρέπει την αξιολόγηση του τρόπου με τον οποίο ο αριθμός επαναλήψεων επηρεάζει την απόδοση του αλγορίθμου. Ο βρόχος ξεκινά με την πρώτη τιμή Kmax =10, και στη συνέχεια προχωρά στις επόμενες τιμές (50,100,200). Για κάθε τιμή N (bits), ο αλγόριθμος εκτελείται για Kmax επαναλήψεις. Αυτό σημαίνει ότι προσαρμόζονται επανειλημμένα τα όρια και τα κέντρα, με στόχο τη βελτίωση της ακρίβειας του κβαντισμού. Αν τα κέντρα δεν αλλάζουν σημαντικά πριν ολοκληρωθούν οι Kmax επαναλήψεις, ο αλγόριθμος τερματίζει νωρίτερα. Μετά το πέρας των Kmax επαναλήψεων, υπολογίζεται το SQNR για την τρέχουσα τιμή του N και αποθηκεύεται στον πίνακα SQNR_results.

```
% Υπολογισμός και σχεδίαση SQNR για διαφορετικά N
SQNR_results = zeros(length(N_bits), length(max_iters)); % Αποθήκευση αποτελεσμάτων

for k_idx = 1:length(max_iters)
    K_max = max_iters(k_idx);
```

Έπειτα εκτελείται ο βρόχος για τον αριθμό bits (N) ο οποίος είναι υπεύθυνος για τον υπολογισμό και την αξιολόγηση της απόδοσης του κβαντισμού για διαφορετικές τιμές του αριθμού bits, που καθορίζουν την ακρίβεια του κβαντιστή. Ο βρόχος διατρέχει κάθε τιμή του πίνακα N_bits και υπολογίζει τον αριθμό των επιπέδων κβαντισμού L, που καθορίζεται από τον αριθμό bits N. Για κάθε bit, ο αριθμός των επιπέδων διπλασιάζεται. Μετά, δημιουργεί έναν πίνακα centers με L ισομερώς κατανεμημένα κέντρα στο διάστημα [min_value,max_value]. Αρχικά, κατανέμονται ομοιόμορφα, αλλά στη συνέχεια θα προσαρμοστούν από τον Lloyd-Max αλγόριθμο. Έπειτα, αρχικοποιεί με τον γνωστό τρόπο τα όρια των περιοχών και αφού αρχικοποιηθούν τα κέντρα και τα όρια, εκτελείται ο Lloyd-Max αλγόριθμος όπως τον έχουμε υλοποιήσει παραπάνω.

```
  for n_idx = 1:length(N_bits)
        N = N_bits(n_idx);
        L = 2^N; % Επίπεδα κβαντισμού

        % Αρχικοποίηση Lloyd-Max
        centers = linspace(min_value, max_value, L); % Ισομερώς κατανεμημένα κέντρα
        thresholds = zeros(1, L + 1);
        thresholds(1) = -Inf;
        thresholds(end) = Inf;
```

```
% Lloyd-Max επαναληπτικός αλγόριθμος
        for iter = 1:K_max
            % Υπολογισμός ορίων βάσει κέντρων
            for i = 2:L
                thresholds(i) = (centers(i-1) + centers(i)) / 2;
            end

            % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
            xq = zeros(size(y));
            for i = 1:L
                indices = y > thresholds(i) & y <= thresholds(i+1);
                xq(indices) = i;
            end

            % Αναπροσαρμογή των κέντρων
            new_centers = zeros(1, L);
            for i = 1:L
                region = y(y > thresholds(i) & y <= thresholds(i+1));
                if ~isempty(region)
                    new_centers(i) = mean(region);
                else
                    new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
                end
            end

            % Έλεγχος για σύγκλιση
            if max(abs(new_centers - centers)) < 1e-10
                break;
            end
            centers = new_centers;
        end
```

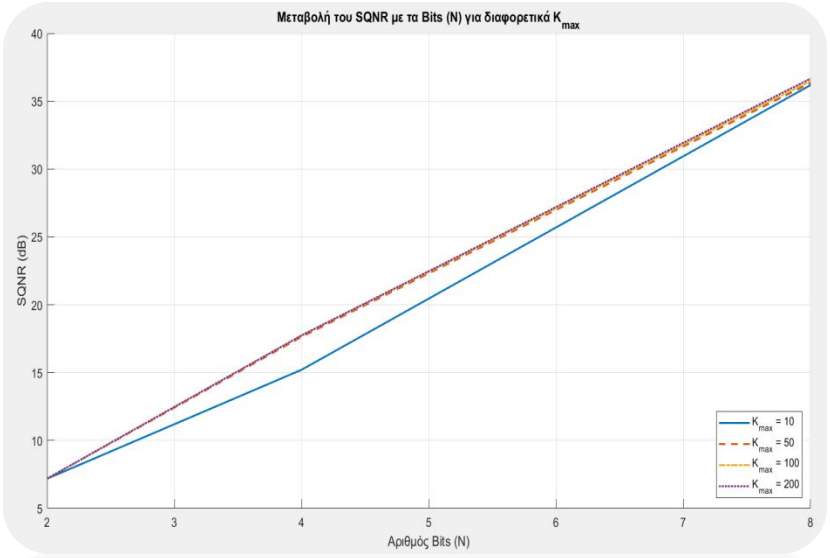
Ανακατασκευάζουμε το σήμα από τις τιμές των κέντρων του κβαντιστή (centers) και τις αντιστοιχίες των επιπέδων (xq). Συγκεκριμένα, η μεταβλητή xq περιέχει τους δείκτες (indices) που αντιστοιχούν στο επίπεδο κβαντισμού για κάθε δείγμα του σήματος. Ο πίνακας centers περιέχει τις τιμές των κέντρων για τα επίπεδα και το αποτέλεσμα είναι το ανακατασκευασμένο σήμα, όπου κάθε δείγμα αντικαθίσταται από το κέντρο του επιπέδου στο οποίο ανήκει. Επίσης, υπολογίζουμε το σφάλμα κβαντισμού (ή αλλιώς θόρυβο), δηλαδή τη διαφορά μεταξύ του αρχικού σήματος y και του ανακατασκευασμένου σήματος reconstructed_signal. Ο θόρυβος κβαντισμού είναι η αιτία της απώλειας πληροφορίας κατά τον κβαντισμό. Θέλουμε ακόμα να υπολογίσουμε την ισχύ του αρχικού σήματος y. Η ισχύς είναι ο μέσος όρος των τετραγώνων των τιμών του σήματος και χρειάζεται γιατί είναι ο αριθμητής στη σχέση του SQNR. Χρησιμοποιείται για να αξιολογηθεί πόση πληροφορία περιέχει το αρχικό σήμα. Αντίστοιχα, ο παρονομαστής στη σχέση του SQNR είναι η ισχύς του θορύβου. Όσο μικρότερη είναι η ισχύς του θορύβου, τόσο καλύτερη είναι η ποιότητα του κβαντισμένου σήματος. Οπότε, υπολογίζουμε και την ισχύ του θορύβου (σφάλματος κβαντισμού), η οποία είναι ο μέσος όρος των τετραγώνων των τιμών του θορύβου noise. Τελικά, υπολογίζεται το SQNR (Signal-to-Quantization Noise Ratio) σε dB, χρησιμοποιώντας τη σχέση:

$$SQNR = 10 \cdot \log_{10}\left(\frac{Signal\ Power}{Noise\ Power}\right)$$

Το SQNR εκφράζει την αναλογία της ισχύος του σήματος προς την ισχύ του θορύβου σε λογαριθμική κλίμακα.

```
% Ανακατασκευή του κβαντισμένου σήματος
        reconstructed_signal = centers(xq);

        % Υπολογισμός SQNR
        noise = y(:) - reconstructed_signal(:); % Θόρυβος
        signal_power = mean(y(:).^2);          % Ισχύς σήματος
        noise_power = mean(noise.^2);          % Ισχύς θορύβου
        SQNR = 10 * log10(signal_power / noise_power); % SQNR σε dB

        % Αποθήκευση του SQNR
        SQNR_results(n_idx, k_idx) = SQNR;
    end
```

Τρέχοντας τον παραπάνω κώδικα λαμβάνουμε το εξής αποτέλεσμα:



ΠΑΡΑΤΗΡΗΣΗ: Το SQNR αυξάνεται καθώς αυξάνεται ο αριθμός των bits N, που είναι αναμενόμενο, καθώς περισσότερα bits κβαντισμού οδηγούν σε μικρότερο σφάλμα κβαντισμού και υψηλότερη αναλογία σήματος προς θόρυβο.Η καμπύλη είναι περίπου γραμμική, κάτι που δείχνει ότι το SQNR αυξάνεται κατά σταθερό ρυθμό για κάθε αύξηση των bits. Ακόμη, για μικρό Kmax (π.χ.,10), το SQNR είναι χαμηλότερο επειδή ο Lloyd-Max δεν έχει συγκλίνει πλήρως. Για μεγαλύτερο Kmax (π.χ.,200), το SQNR βελτιώνεται, καθώς ο Lloyd-Max έχει επαναληφθεί περισσότερες φορές, επιτυγχάνοντας καλύτερη κατανομή των κέντρων κβαντισμού. Οι καμπύλες συγκλίνουν καθώς αυξάνονται τα bits, που σημαίνει ότι για μεγάλα N, οι διαφορές από Kmax γίνονται λιγότερο σημαντικές.

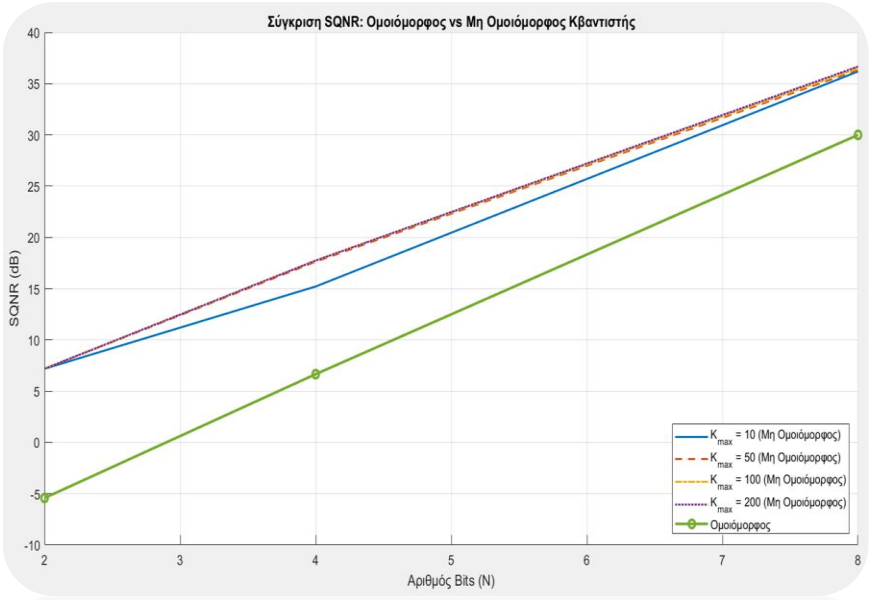
(ii). Τώρα καλούμαστε να κάνουμε την ίδια σύγκριση αν επιπλέον χρησιμοποιήσουμε το σχήμα (a) (ομοιόμορφη κβάντιση). Αφού ακολουθήσουμε την ίδια διαδικασία για φόρτωση του αρχείου ήχου, κανονικοποίηση του σήματος στο [-1,1] και ορισμό παραμέτρων, πρώτα δημιουργούμε δύο πίνακες για την αποθήκευση των αποτελεσμάτων του SQNR:
SQNR_results_b: Αποθηκεύει τα αποτελέσματα για διαφορετικά N και Kmax
SQNR_results_a: Αποθηκεύει τα αποτελέσματα του SQNR για το σχήμα (a), που αναφέρεται στον ομοιόμορφο κβαντιστή (για διάφορες τιμές N).
Χρησιμοποιούνται για την αποθήκευση των τιμών SQNR που θα υπολογιστούν και για τη δημιουργία των αντίστοιχων γραφημάτων. Ο βρόχος για τον υπολογισμό του σχήματος (a) ξεκινά διατρέχοντας όλες τις τιμές του πίνακα N_bits, δηλαδή τους διαφορετικούς αριθμούς bits N). Κάθε τιμή N αντιστοιχεί σε διαφορετικό αριθμό επιπέδων κβαντισμού L=2^N, και πρέπει να υπολογιστεί το SQNR για κάθε μία από αυτές. Έπειτα, υπολογίζει τον αριθμό των επιπέδων κβαντισμού (L) με βάση τον αριθμό bits και το βήμα κβαντισμού (Δ), που είναι η διαφορά μεταξύ δύο διαδοχικών επιπέδων, ενώ συνεχίζει υπολογίζοντας και τα κέντρα κβαντισμού, δηλαδή τις διακριτές τιμές που αντιστοιχούν σε κάθε επίπεδο κβαντισμού. Αυτά είναι ισομερώς κατανεμημένα και τοποθετημένα στο μέσο κάθε περιοχής όπως έχουμε αναφέρει και παραπάνω.

```
% Υπολογισμός και σχεδίαση SQNR για διαφορετικά N
SQNR_results_b = zeros(length(N_bits), length(max_iters)); % Αποθήκευση
αποτελεσμάτων για σχήμα (b)
SQNR_results_a = zeros(1, length(N_bits));                 % Αποθήκευση
αποτελεσμάτων για σχήμα (a)

% Υπολογισμός για σχήμα (a) - Ομοιόμορφος Κβαντιστής
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers = linspace(min_value + delta/2, max_value - delta/2, L);
```

Μετά κάνουμε κβάντηση στο σήμα, δηλαδή αντιστοιχίζουμε κάθε δείγμα του σήματος y σε ένα επίπεδο κβαντισμού με την εξής διαδικασία. Υπολογίζουμε σε ποια περιοχή ανήκει το δείγμα ($\frac{y-\min\_value}{\Delta}$), και στρογγυλοποιούμε το αποτέλεσμα για να βρει το πλησιέστερο επίπεδο. Προσθέτουμε +1 για να μετατρέψει τον δείκτη σε βάση 1 (MATLAB). Χρησιμοποιούμε τη min και max για να περιορίσει τους δείκτες εντός του εύρους [1,L]. Αφού ολοκληρωθεί αυτό ανακατασκευάζουμε το κβαντισμένο σήμα, αντικαθιστώντας κάθε δείγμα του σήματος y με το αντίστοιχο κέντρο του επιπέδου στο οποίο ανήκει (xq). Όμοια με το παραπάνω ερώτημα υπολογίζουμε ακολούθως τον θόρυβο κβαντισμού, την ισχύ του σήματος και την ισχύ του θορύβου και υπολογίζουμε το SQNR σε dB. Τέλος, αποθηκεύουμε την τιμή του SQNR στον πίνακα SQNR_results_a για την τρέχουσα τιμή των bits (N).

```
xq = min(max(round((y - min_value) / delta) + 1, 1), L);
    reconstructed_signal = centers(xq);

    % Υπολογισμός SQNR
    noise = y(:) - reconstructed_signal(:); % Θόρυβος
    signal_power = mean(y(:).^2);          % Ισχύς σήματος
    noise_power = mean(noise.^2);          % Ισχύς θορύβου
    SQNR = 10 * log10(signal_power / noise_power); % SQNR σε dB

    % Αποθήκευση αποτελεσμάτων
    SQNR_results_a(n_idx) = SQNR;
end
```

Τρέχοντας τον παραπάνω κώδικα λαμβάνουμε το εξής αποτέλεσμα:



ΠΑΡΑΤΗΡΗΣΗ: Η καμπύλη του ομοιόμορφου κβαντιστή (πράσινη γραμμή) δείχνει σαφώς χαμηλότερες τιμές SQNR συγκριτικά με τον μη ομοιόμορφο κβαντιστή. Το SQNR αυξάνεται με τον αριθμό bits (N), αλλά ο ρυθμός αύξησης είναι μικρότερος σε σχέση με τον μη ομοιόμορφο κβαντιστή. Η απόδοσή του δεν επηρεάζεται από τις επαναλήψεις (Kmax), καθώς η προσέγγιση του ομοιόμορφου κβαντιστή δεν περιλαμβάνει δυναμική προσαρμογή. Άρα, ο μη ομοιόμορφος κβαντιστής αποδίδει καλύτερα λόγω της προσαρμογής των επιπέδων μέσω του Lloyd-Max, ενώ ο ομοιόμορφος βασίζεται σε σταθερό βήμα.

ΣΧΟΛΙΟ ΓΙΑ ΤΙΣ ΑΠΟΔΟΣΕΙΣ:
Για μικρό N (N=2), το SQNR είναι αρνητικό (π.χ., −5 dB), γεγονός που δείχνει υψηλό θόρυβο. Για μεγαλύτερο N (N=8), το SQNR αυξάνεται (~30 dB), αλλά παραμένει χαμηλότερο από τον μη ομοιόμορφο.Από την άλλη, για N=2, το SQNR είναι αισθητά υψηλότερο από τον ομοιόμορφο, π.χ.,(~10 dB) και για N=8, το SQNR φτάνει ~35 dB, ξεπερνώντας τον ομοιόμορφο. Η αύξηση του Kmax οδηγεί σε μικρή βελτίωση του SQNR, καθώς ο αλγόριθμος συγκλίνει γρήγορα στις πρώτες επαναλήψεις. Συμπαιρένουμε πως ο Lloyd-Max είναι προτιμότερος για εφαρμογές που απαιτούν υψηλή ποιότητα αναπαράστασης, ιδιαίτερα για μικρά N.

(iii). Πρέπει να υλοποιήσουμε και να συγκρίνουμε δύο διαφορετικούς κβαντιστές (ομοιόμορφο και Lloyd-Max) για ένα αρχικό σήμα ήχου το οποίο είναι χωρίς κωδικοποίηση, δηλαδή δεν εφαρμόζεται καμία διαδικασία μετατροπής ή απώλειας πληροφορίας στο αρχικό σήμα.Το αρχικό σήμα χρησιμεύει ως σημείο αναφοράς για τη σύγκριση με τα ανακατασκευασμένα σήματα. Μετά την ανακατασκευή τα αναπαράγουμε για να αξιολογηθεί η ποιότητα τους. Φορτώνουμε το αρχικό ηχητικό αρχείο speech.wav και εξάγουμε το σήμα y και τη συχνότητα δειγματοληψίας fs. Κανονικοποιούμε το σήμα έτσι ώστε να έχει τιμές εντός του διαστήματος [−1,1]. Αυτό είναι απαραίτητο για τη σωστή λειτουργία του κβαντιστή. Μετά καθορίζουμε τη δυναμική περιοχή του σήματος ([−1,1]), τους αριθμούς bits κβαντισμού που θα εξεταστούν (N=2,4,8), τον μέγιστο αριθμό επαναλήψεων για τον αλγόριθμο Lloyd-Max (Kmax =100),το όριο σύγκλισης (ϵ=10 ^(−6)) και τον χρονικό άξονα t, που χρησιμοποιείται για τη σχεδίαση των κυματομορφών.

```
% Φόρτωση του αρχικού αρχείου ήχου
[y, fs] = audioread('speech.wav');
y = y / max(abs(y));                % Κανονικοποίηση του σήματος στο [-1, 1]

% Παράμετροι
min_value = -1;                     % Ελάχιστη τιμή
max_value = 1;                      % Μέγιστη τιμή
N_bits = [2, 4, 8];                 % Bits κβαντισμού
K_max = 100;                        % Επαναλήψεις για τον Lloyd-Max
epsilon = 1e-6;                     % Όριο σύγκλισης για τον Lloyd-Max
t = (0:length(y)-1) / fs;           % Χρονικός άξονας
```

Στη συνέχεια υλοποιείται ξανά βρόχος που διατρέχει όλες τις τιμές του N που ορίζονται από τον πίνακα N_bits ([2,4,8]) και για κάθε τιμή N υπολογίζονται τα επίπεδα κβαντισμού L και εφαρμόζονται οι δύο κβαντιστές (ομοιόμορφος και Lloyd-Max). Επαναλαμβάνουμε τη διαδικασία για κάθε στοιχείο του πίνακα N_bits. Η μεταβλητή n_idx είναι ο δείκτης που αντιπροσωπεύει την τρέχουσα θέση στον πίνακα και ακριβώς μετά γίνεται η εκχώρηση της τρέχουσας τιμής του N στη μεταβλητή N. Για παράδειγμα όταν n_idx=1, N=2. Αφου ολοκληρωθεί ο βρόχος, περνάμε στον ομοιόμορφο κβαντιστή, ο οποίος διαχωρίζει τη δυναμική περιοχή σε ίσα διαστήματα και αναπαριστά το σήμα με σταθερά επίπεδα. Κατά τα γνωστά πρώτα υπολογίζουμε το βήμα κβαντισμού Δ, δηλαδή το διάστημα μεταξύ δύο διαδοχικών επιπέδων, μετά δημιουργουμε έναν πίνακα centers_uniform που περιέχει τα κέντρα των επιπέδων κβαντισμού (τα κέντρα είναι ισαπέχουσες τιμές μέσα στη δυναμική περιοχή [min_value,max_value], τοποθετημένα στο μέσο κάθε περιοχής.) και μετά υπολογίζουμε σε ποιο επίπεδο ανήκει κάθε δείγμα του σήματος y υπολογίζοντας τη θέση του δείγματος στη δυναμική περιοχή, διαιρώντας τη διαφορά y−min_value με το Δ και στρογγυλοποιώντας τη θέση προς τα πάνω, χρησιμοποιώντας τη συνάρτηση ceil. Σημαντικό είναι να διορθώσουμε τους δείκτες ώστε να είναι εντός του επιτρεπτού εύρους, πιο στγκεκριμένα αν ένας δείκτης είναι 0 (όταν το δείγμα είναι στο min_value), τον θέτει σε 1, ενώ αν ένας δείκτης είναι μεγαλύτερος από L (όταν το δείγμα είναι στο max_value), τον θέτει σε L. Τέλος, ανακατασκευάζουμε το σήμα αντικαθιστώντας κάθε δείγμα με το αντίστοιχο κέντρο του επιπέδου στο οποίο ανήκει.

```
% Υπολογισμός και σχεδίαση για κάθε N
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού

    %% Ομοιόμορφος Κβαντιστής
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers_uniform = linspace(min_value + delta/2, max_value - delta/2, L); %
Κέντρα
    xq_uniform = ceil((y - min_value) / delta);
    xq_uniform(xq_uniform == 0) = 1;    % Αν κάποιο δείγμα είναι στο min_value
    xq_uniform(xq_uniform > L) = L;     % Αν κάποιο δείγμα είναι στο max_value
    reconstructed_uniform = centers_uniform(xq_uniform); % Ανακατασκευή
```

Έπειτα υλοποιούμε τον μη ομοιόμορφο κβαντιστή κατά τα γνωστά με την χρήση του Lloyd_Max. Δημιουργούμε τον πίνακα centers με L ισαπέχουσες τιμές (τα αρχικά κέντρα κβαντισμού) εντός της δυναμικής περιοχής [min_value,max_value], έπειτα τον πίνακα thresholds για τα όρια περιοχών του σήματος με το πρώτο όριο να ορίζεται −∞ και το τελευταίο +∞, ώστε να περιλαμβάνεται όλο το σήμα και ύστερα εφαρμόζεται ο επαναληπτικός αλγόριθμος ακριβώς με τον ίδιο τρόπο όπως και στα προηγούμενα ερωτήματα.

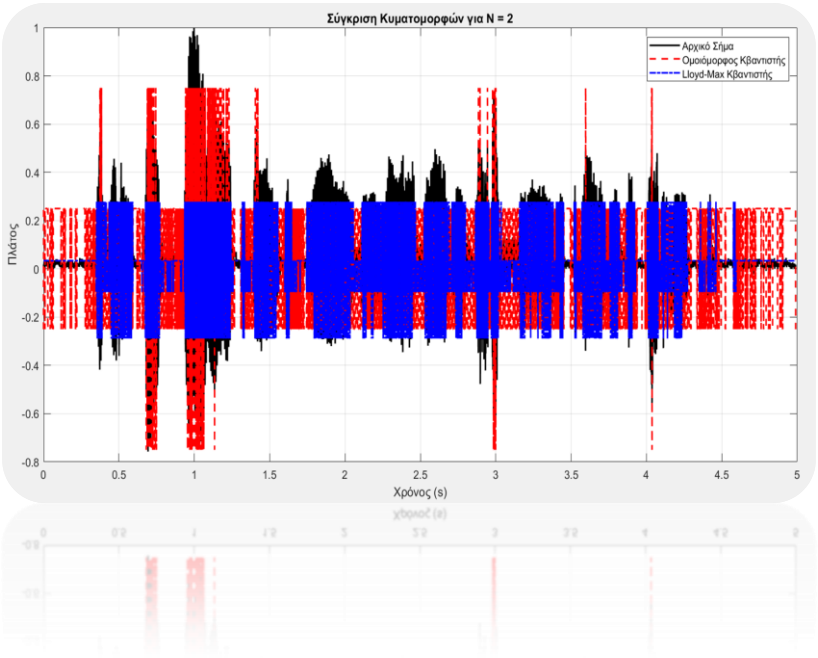
```
%% Lloyd-Max Κβαντιστής
    centers = linspace(min_value, max_value, L); % Αρχικά κέντρα
    thresholds = zeros(1, L + 1);
    thresholds(1) = -Inf;
    thresholds(end) = Inf;

    % Lloyd-Max επαναληπτικός αλγόριθμος
    for iter = 1:K_max
        % Υπολογισμός ορίων βάσει κέντρων
        for i = 2:L
            thresholds(i) = (centers(i-1) + centers(i)) / 2;
        end

        % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
        xq = zeros(size(y));
        for i = 1:L
            indices = y > thresholds(i) & y <= thresholds(i+1);
            xq(indices) = i;
        end

        % Αναπροσαρμογή των κέντρων
        new_centers = zeros(1, L);
        for i = 1:L
            region = y(y > thresholds(i) & y <= thresholds(i+1));
            if ~isempty(region)
                new_centers(i) = mean(region);
            else
                new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
            end
        end

        % Έλεγχος για σύγκλιση
        if max(abs(new_centers - centers)) < epsilon
            break;
        end
        centers = new_centers;
    end

    % Ανακατασκευή του σήματος
    reconstructed_lloyd_max = centers(xq);
```

Τέλος, κάνουμε την αναπαραγωγή των σημάτων. Αναπαράγουμε το αρχικό σήμα y χρησιμοποιώντας τη συνάρτηση sound, η οποία στέλνει τα δεδομένα του σήματος στον ήχο (ηχεία ή ακουστικά).Το σήμα αναπαράγεται με τη συχνότητα δειγματοληψίας fs, που έχει ήδη οριστεί κατά τη φόρτωση του αρχείου. Μετά την κάθε αναπαραγωγή σταματάμε την εκτέλεση του κώδικα για ένα χρονικό διάστημα ίσο με τη διάρκεια του σήματος συν ένα επιπλέον δευτερόλεπτο ώστε ο χρήστης να έχει χρόνο να ακούσει το τέλος του σήματος πριν την αναπαραγωγή του επόμενου. Πρώτα κάνουμε αναπαραγωγή του σήματος reconstructed_uniform, το οποίο έχει κβαντιστεί και ανακατασκευαστεί χρησιμοποιώντας τον ομοιόμορφο κβαντιστή και μετά του σήματος reconstructed_lloyd_max, το οποίο έχει κβαντιστεί και ανακατασκευαστεί χρησιμοποιώντας τον Lloyd-Max κβαντιστή.

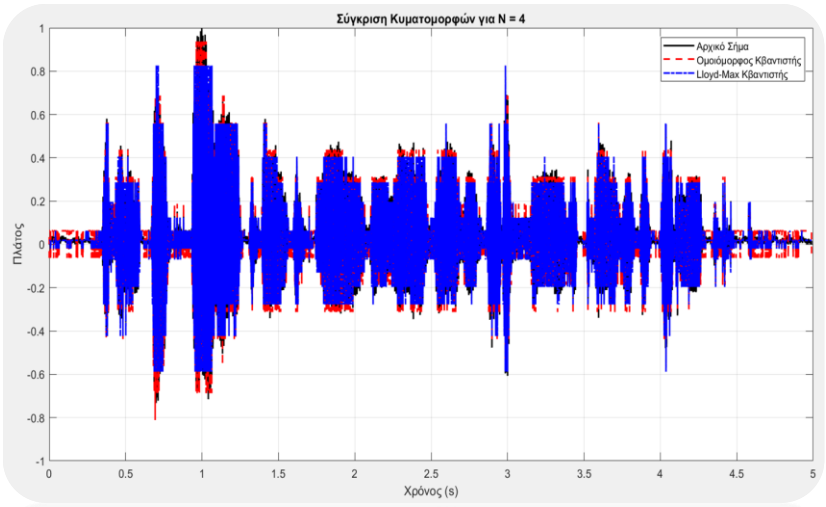
```
%% Αναπαραγωγή Σημάτων
    disp(['Αναπαραγωγή του αρχικού σήματος για N = ', num2str(N)]);
    sound(y, fs);
    pause(length(y) / fs + 1);

    disp(['Αναπαραγωγή του ομοιόμορφου σήματος για N = ', num2str(N)]);
    sound(reconstructed_uniform, fs);
    pause(length(y) / fs + 1);

    disp(['Αναπαραγωγή του Lloyd-Max σήματος για N = ', num2str(N)]);
    sound(reconstructed_lloyd_max, fs);
    pause(length(y) / fs + 1);
end
```

Αφού τρέξουμε όλο τον κώδικα θα έχουμε τα εξής αποτελέσματα:

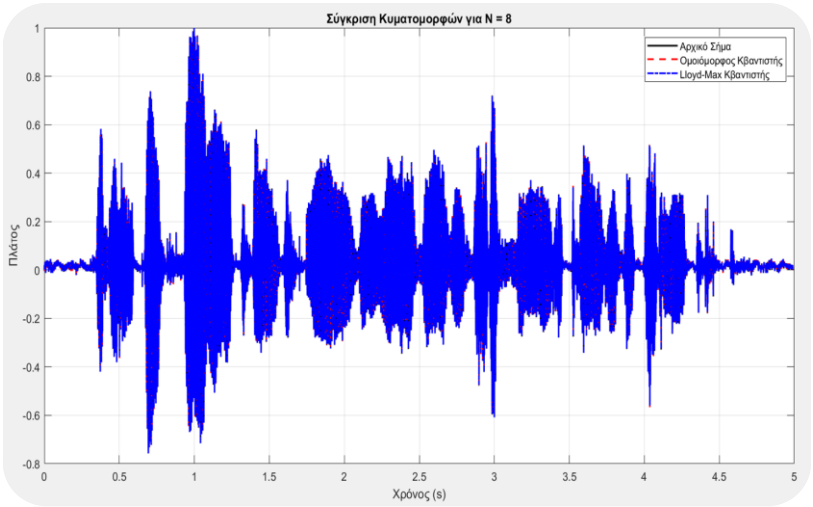
ΠΑΡΑΤΗΡΗΣΗ:



Για N = 2: Ο ομοιόμορφος κβαντιστής εμφανίζει μεγάλες αποκλίσεις από το αρχικό σήμα, καθώς τα 4 επίπεδα κβαντισμού (L=2 ^2=4) είναι λίγα για να αναπαραστήσουν την κυματομορφή με ακρίβεια, οπότε ο θόρυβος κβαντισμού είναι πολύ έντονος. Ο μη ομοιόμορφος κβαντιστής παρότι έχει επίσης 4 επίπεδα κβαντισμού, οι αποκλίσεις από το αρχικό σήμα είναι μικρότερες. Η δυναμική προσαρμογή των επιπέδων στα δεδομένα βοηθά στη μείωση του θορύβου κβαντισμού.



Για N = 4: Παρατηρείται σαφής βελτίωση σε σχέση με την περίπτωση N=2 για τον ομοιόμορφο κβαντιστή, καθώς ο αριθμός των επιπέδων κβαντισμού αυξάνεται σε L=2 ^4 =16. Οι αποκλίσεις είναι ακόμα εμφανείς, ειδικά σε περιοχές υψηλής πυκνότητας του σήματος. Για τον μη ομοιόμορφο κβαντιστή η ποιότητα βελτιώνεται αισθητά σε σύγκριση με τον ομοιόμορφο κβαντιστή. Οι αποκλίσεις είναι ελάχιστες, ιδιαίτερα στις περιοχές υψηλής πυκνότητας.



Για N = 8: Με L=2 ^8 =256 επίπεδα κβαντισμού, η αναπαράσταση του σήματος με τον ο οιόμορφο κβαντιστή είναι σχεδόν ταυτόσημη με το αρχικό. Οι αποκλίσεις είναι ελάχιστες, αλλά παραμένουν ελαφρώς μεγαλύτερες από τον Lloyd-Max. Για τον μη ομοιόμορφο κβαντιστή η ποιότητα είναι σχεδόν πανομοιότυπη με το αρχικό σήμα. Η προσαρμογή των επιπέδων εξασφαλίζει την ελάχιστη απόκλιση, ακόμα και με περιορισμένο αριθμό bits.

Συμπεραίνουμε ότι ο ομοιόμορφος κβαντιστής είναι πιο απλός, αλλά δεν είναι τόσο αποδοτικός για χαμηλές τιμές N ενώ ο μη ομοιόμορφος είναι πιο αποδοτικός, ειδικά όταν ο αριθμός των bits είναι περιορισμένος, και μπορεί να δώσει καλύτερη ποιότητα σήματος για δεδομένο αριθμό bits.

(iv). Σε αυτό το ερώτημα σκοπός μας είναι να υπολογίσουμε το μέσο τετραγωνικό σφάλμα (MSE) για τον ομοιόμορφο και τον μη ομοιόμορφο κβαντιστή για διαφορετικές τιμές N (bits κβαντισμού). Για να το πετύχουμε αυτό πρέπει να δημιουργήσουμε δύο πίνακες (MSE_uniform και MSE_lloyd_max) για την αποθήκευση του MSE για κάθε τιμή N. Οι πίνακες έχουν μήκος ίσο με τον αριθμό των τιμών στο N_bits, δηλαδή length(N_bits) και αρχικοποιούνται με μηδενικές τιμές. Οι πίνακες αυτοί θα χρησιμοποιηθούν για να αποθηκευτούν τα αποτελέσματα του MSE (ένα για κάθε τιμή N) για τους δύο κβαντιστές.

```
% Αρχικοποίηση αποθήκευσης MSE
MSE_uniform = zeros(1, length(N_bits)); % MSE για τον ομοιόμορφο
MSE_lloyd_max = zeros(1, length(N_bits)); % MSE για τον Lloyd-Max
```

Δημιουργούμε ξανά τον βρόχο ο οποίος θα διατρέχει όλες τις τιμές του N που βρίσκονται στον πίνακα N_bits. Σε κάθε επανάληψη θα ενημερώνει τη μεταβλητή N με την τρέχουσα τιμή του αριθμού των bits και θα υπολογίζει τον αριθμό των επιπέδων κβαντισμού L, τα οποία εξαρτώνται από το N. Δημιουργούμε τον ομοιόμορφο κβαντιστή όπως έχουμε κάνει και παραπάνω και για αυτόν υπολογίζουμε το σφάλμα κβαντισμού (ή "θόρυβο κβαντισμού").
Η διαφορά υπολογίζεται στοιχείο προς στοιχείο ανάμεσα στο αρχικό σήμα y και στο ανακατασκευασμένο σήμα reconstructed_uniform που προκύπτει από τον ομοιόμορφο κβαντιστή. Το (:) μετατρέπει τον πίνακα σε στήλη, διασφαλίζοντας ότι τα δεδομένα έχουν κατάλληλες διαστάσεις για τον υπολογισμό της διαφοράς. Έτσι, δημιουργείται ένας πίνακας error_uniform που περιέχει το σφάλμα για κάθε δείγμα. Συνεχίζουμε τώρα υπολογίζοντας το μέσο τετραγωνικό σφάλμα (MSE) για τον ομοιόμορφο κβαντιστή, χρησιμοποιώντας τη διαφορά σήματος error_uniform. Κάθε στοιχείο του error_uniform υψώνεται στο τετράγωνο:

error_uniform ^2 =[e1 ^2 ,e2 ^2 ,… ,eN ^2]

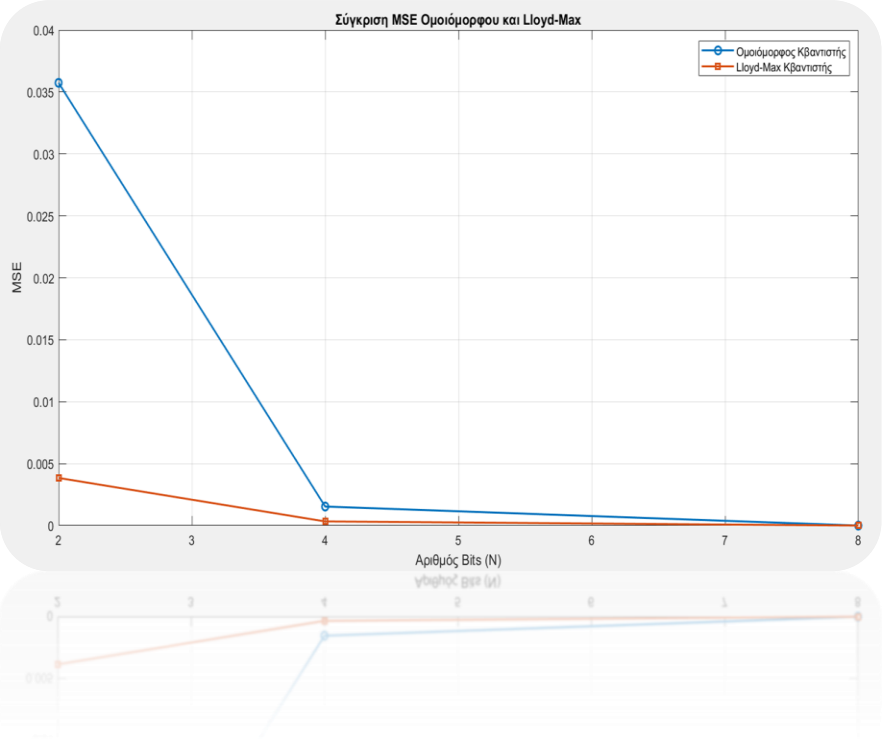
Όπου, ei =y[i] − reconstructed_uniform[i]. Έπειτα Υπολογίζεται ο μέσος όρος των τετραγώνων των διαφορών: MSE=$\frac{1}{N}\sum_{i=1}^{N} ei$ ^2, όπου N είναι ο αριθμός των δειγμάτων. Το αποτέλεσμα αποθηκεύεται στη θέση n_idx του πίνακα MSE_uniform, που αντιστοιχεί στην τρέχουσα τιμή N. Το MSE είναι ένα μέτρο που δείχνει πόσο καλά προσεγγίζει ο κβαντιστής το αρχικό σήμα. Όσο μικρότερο είναι το MSE, τόσο πιο κοντά είναι το ανακατασκευασμένο σήμα στο αρχικό.

```
% Υπολογισμός MSE για κάθε τιμή N
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού

    %% Ομοιόμορφος Κβαντιστής
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers_uniform = linspace(min_value + delta/2, max_value - delta/2, L); % Κέντρα
    xq_uniform = ceil((y - min_value) / delta);
    xq_uniform(xq_uniform == 0) = 1;    % Αν κάποιο δείγμα είναι στο min_value
    xq_uniform(xq_uniform > L) = L;     % Αν κάποιο δείγμα είναι στο max_value
    reconstructed_uniform = centers_uniform(xq_uniform); % Ανακατασκευή

    % Υπολογισμός MSE για τον ομοιόμορφο
    error_uniform = y(:) - reconstructed_uniform(:);
    MSE_uniform(n_idx) = mean(error_uniform.^2);
```

Ακολουθούμε ακριβώς την ίδια διαδικασία και για τον μη ομοιόμορφο κβαντιστή.

```
%% Lloyd-Max Κβαντιστής
    centers = linspace(min_value, max_value, L); % Αρχικά κέντρα
    thresholds = zeros(1, L + 1);
    thresholds(1) = -Inf;
    thresholds(end) = Inf;

    % Lloyd-Max επαναληπτικός αλγόριθμος
    for iter = 1:K_max
        % Υπολογισμός ορίων βάσει κέντρων
        for i = 2:L
            thresholds(i) = (centers(i-1) + centers(i)) / 2;
        end

        % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
        xq = zeros(size(y));
        for i = 1:L
            indices = y > thresholds(i) & y <= thresholds(i+1);
            xq(indices) = i;
        end

        % Αναπροσαρμογή των κέντρων
        new_centers = zeros(1, L);
        for i = 1:L
            region = y(y > thresholds(i) & y <= thresholds(i+1));
            if ~isempty(region)
                new_centers(i) = mean(region);
            else
                new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
            end
        end

        % Έλεγχος για σύγκλιση
        if max(abs(new_centers - centers)) < epsilon
            break;
        end
        centers = new_centers;
    end

    % Ανακατασκευή του σήματος
    reconstructed_lloyd_max = centers(xq);

    % Υπολογισμός MSE για τον Lloyd-Max
    error_lloyd_max = y(:) - reconstructed_lloyd_max(:);
    MSE_lloyd_max(n_idx) = mean(error_lloyd_max.^2);
end
```

Και τρέχοντας τον παραπάνω κώδικα θα παρατηρήσουμε το εξής αποτέλεσμα:



ΠΑΡΑΤΗΡΗΣΗ:

Ομοιόμορφος Κβαντιστής:
Το MSE είναι μεγαλύτερο για μικρές τιμές N, αλλά μειώνεται σημαντικά καθώς αυξάνεται το N. Η απόδοση βελτιώνεται, αλλά παραμένει χαμηλότερη από τον Lloyd-Max.

Μη Ομοιόμορφος Κβαντιστής:
Το MSE είναι αισθητά χαμηλότερο για όλες τις τιμές N, καθώς ο Lloyd-Max κβαντιστής προσαρμόζει τα επίπεδα κβαντισμού στις πυκνές περιοχές του σήματος.Το σφάλμα μειώνεται γρηγορότερα και παραμένει πιο σταθερό για υψηλές τιμές N.

Για μικρές Τιμές N (π.χ., N=2) ο ομοιόμορφος κβαντιστής έχει υψηλό MSE (~0.035). Αυτό οφείλεται στο γεγονός ότι τα λίγα επίπεδα κβαντισμού (L=4) δεν επαρκούν για την ακριβή αναπαράσταση του σήματος.
Ο Lloyd-Max κβαντιστής έχει πολύ μικρότερο MSE (~0.005). Η προσαρμογή των επιπέδων του μειώνει σημαντικά το σφάλμα. Για μεσαίες Τιμές N (π.χ.,N=4) ο ομοιόμορφος κβαντιστής παρουσιάζει σημαντική βελτίωση (MSE ~0.005). Ο μη ομοιόμορφος εξακολουθεί να υπερέχει, αν και η διαφορά είναι μικρότερη.Για μεγάλες Τιμές N (π.χ., N=8) το MSE για τον ομοιόμορφο και τον μη ομοιόμορφο κβαντιστή είναι σχεδόν μηδενικό. Αυτό συμβαίνει επειδή τα L=2 ^8=256 επίπεδα κβαντισμού επαρκούν για να αναπαραστήσουν το σήμα με πολύ μικρή απώλεια.

Ο μη ομοιόμορφος κβαντιστής είναι πιο αποδοτικός για μικρό αριθμό bits N, λόγω της προσαρμοστικότητάς του στις κατανομές του σήματος. Αυτή η απόδοση είναι ιδιαίτερα σημαντική σε εφαρμογές όπου ο αριθμός των bits είναι περιορισμένος (π.χ., σε συστήματα με περιορισμένο bandwidth). Καθώς αυξάνεται το N, και οι δύο κβαντιστές συγκλίνουν σε πολύ μικρό MSE, καθώς περισσότερα επίπεδα κβαντισμού επιτρέπουν την πιο ακριβή αναπαράσταση του σήματος.

A2. Στα πειράματα που θα εκτελέσετε η δυναμική περιοχή του κβαντιστή να είναι μεταξύ των τιμών max_value = 3.5 , min_value = −3.5 .
(i). Να υλοποιήσετε το παραπάνω σύστημα κωδικοποίησης/αποκωδικοποίησης DPCM.
(ii). Επιλέξτε δύο τιμές του $p \geq 5$ και για N = 1,2,3 bits σχεδιάστε στο ίδιο γράφημα το αρχικό σήμα και το σφάλμα πρόβλεψης y. Σχολιάστε τα αποτελέσματα. Τί παρατηρείτε;
(iii). Αξιολογείστε την απόδοσή του με γράφημα που να δείχνει το μέσο τετραγωνικό σφάλμα πρόβλεψης ως προς το N και για διάφορες τιμές του p. Συγκεκριμένα, για πλήθος δυαδικών ψηφίων N = 1,2,3bits τα οποία χρησιμοποιεί ο ομοιόμορφος κβαντιστής για την κωδικοποίηση του σήματος πρόβλεψης και για τάξη προβλέπτη p = 5: 10. Επιπλέον, για κάθε p καταγράψτε στην αναφορά σας και σχολιάστε τις τιμές των συντελεστών του προβλέπτη.
(iv). Για N = 1,2,3bits να απεικονίσετε το αρχικό και το ανακατασκευασμένο σήμα στο δέκτη για p = 5,10 και να σχολιάσετε τα αποτελέσματα της ανακατασκευής σε σχέση με τα bits κβάντισης.

(i). Για αρχή φορτώνουμε το αρχείο source.mat, το οποίο περιέχει το σήμα εισόδου t. Η μεταβλητή t περιέχει τα δείγματα του αρχικού σήματος που θα κωδικοποιηθεί και αποκωδικοποιηθεί. Ορίζουμε πάλι τις παραμέτρους N, min_value, max_value, delta και αρχικοποιούμε τους μηδενικούς πίνακες:
quantized_signal: Αποθηκεύει το κβαντισμένο σήμα.
reconstructed_signal: Αποθηκεύει το ανακατασκευασμένο σήμα.
error_signal: Αποθηκεύει το σφάλμα πρόβλεψης.
Η num_samples ορίζει τον αριθμό δειγμάτων του σήματος t(το συνολικό μήκος του σήματος). Οι quantized_signal, reconstructed_signal και error_signal είναι μονοδιάστατοι πίνακες στήλης, οι οποίοι αντιστοιχούν σε κάθε δείγμα του σήματος t.

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
N = 8; % Αριθμός bits του κβαντιστή
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής
quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

% Αρχικοποίηση
num_samples = length(t); % Χρήση της μεταβλητής t
quantized_signal = zeros(num_samples, 1);
reconstructed_signal = zeros(num_samples, 1);
error_signal = zeros(num_samples, 1);
```

Τώρα υλοποιούμε τον κβαντισμό των συντελεστών πρόβλεψης που χρησιμοποιούνται για την πρόβλεψη στο φίλτρο DPCM. Ξεκινάμε ορίζοντας τους αρχικούς συντελεστές πρόβλεψης (πριν τον κβαντισμό). Εδώ στην περίπτωσή μας, το φίλτρο πρόβλεψης έχει μόνο έναν συντελεστή, τον 0.9, που αντιστοιχεί σε φίλτρο πρόβλεψης 1ου βαθμού. Αυτό σημαίνει ότι η πρόβλεψη βασίζεται στο 90% του προηγούμενου δείγματος. Μετά, ορίζουμε τη δυναμική περιοχή των συντελεστών πρόβλεψης, που είναι το διάστημα [−2,2]. Οι συντελεστές πρόβλεψης πρέπει να περιοριστούν σε αυτή τη δυναμική περιοχή, ώστε να είναι συμβατοί με το σύστημα.Αυτή η περιοχή επιλέγεται επειδή το σύστημα DPCM χρησιμοποιεί έναν ομοιόμορφο κβαντιστή για να κβαντίσει τις τιμές των συντελεστών πρόβλεψης, και αυτή η περιοχή προσφέρει επαρκή ακρίβεια για τους συντελεστές. Ακόμα, υπολογίζουμε το βήμα κβαντισμού (Δ) για τους συντελεστές πρόβλεψης και εκτελούμε τον κβαντισμό των συντελεστών πρόβλεψης. Συγκεκριμένα, πρώτα διαιρούμε τους αρχικούς συντελεστές με το βήμα κβαντισμού delta_coefs, μετατρέποντας την τιμή σε "κλίμακα επιπέδων". Στρογγυλοποιούμε το αποτέλεσμα στο πλησιέστερο επίπεδο κβαντισμού. Ως επόμενο βήμα έχουμε την επιστροφή την τιμή από την "κλίμακα επιπέδων" πίσω στη δυναμική περιοχή, πολλαπλασιάζοντας με το delta_coeffs. Τελευταίο βήμα είναι η εξασφάλιση ότι η κβαντισμένη τιμή δεν υπερβαίνει τη μέγιστη τιμή max_value_coeffs=2 και ότι η κβαντισμένη τιμή δεν είναι μικρότερη από την ελάχιστη τιμή min_value_coeffs=−2. Αν η κβαντισμένη τιμή ήταν 2.1, ο περιορισμός θα την έφερνε στο 2. Η τελική τιμή του κβαντισμένου συντελεστή αποθηκεύεται στη μεταβλητή predictor_coeffs.

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
N = 8; % Αριθμός bits του κβαντιστή
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής
quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή
```

```
% Αρχικοποίηση
num_samples = length(t); % Χρήση της μεταβλητής t
quantized_signal = zeros(num_samples, 1);
reconstructed_signal = zeros(num_samples, 1);
error_signal = zeros(num_samples, 1);

% Συντελεστές φίλτρου πρόβλεψης (πριν τον κβαντισμό)
predictor_coeffs_original = [0.9];

% Παράμετροι του ομοιόμορφου κβαντιστή
min_value_coeffs = -2; % Ελάχιστη τιμή δυναμικής περιοχής
max_value_coeffs = 2;  % Μέγιστη τιμή δυναμικής περιοχής
delta_coeffs = (max_value_coeffs - min_value_coeffs) / (2^8); % Βήμα κβαντισμού

% Κβαντισμός των συντελεστών πρόβλεψης
predictor_coeffs = max(min(round(predictor_coeffs_original / delta_coeffs) *
delta_coeffs, max_value_coeffs), min_value_coeffs);
```

Μετά υπολογίζουμε την τάξη του φίλτρου πρόβλεψης. Το φίλτρο πρόβλεψης έχει ως συντελεστές τους predictor_coeffs. Ο αριθμός των συντελεστών καθορίζει πόσα προηγούμενα δείγματα χρησιμοποιούνται για την πρόβλεψη του τρέχοντος δείγματος. Η length(predictor_coeffs) επιστρέφει το μέγεθος του πίνακα predictor_coeffs, δηλαδή τον αριθμό των συντελεστών πρόβλεψης. Αν δηλαδή predictor_coeffs = [0.9, 0.1]; τότε predictor_order = length(predictor_coeffs); % Τάξη = 2. Αυτό σημαίνει ότι το φίλτρο πρόβλεψης χρησιμοποιεί 2 προηγούμενα δείγματα. Ορίζουμε μια ανώνυμη συνάρτηση για τον υπολογισμό της πρόβλεψης ενός δείγματος βασισμένου στα προηγούμενα δείγματα με παραμέτρους:
coeffs: Οι συντελεστές πρόβλεψης (predictor_coeffs).
signal: Το σήμα (reconstructed_signal ή decoded_signal).
n: Η θέση του τρέχοντος δείγματος που θέλουμε να προβλέψουμε.
order: Η τάξη του φίλτρου πρόβλεψης (predictor_order), δηλαδή πόσα προηγούμενα δείγματα θα χρησιμοποιηθούν.
Αυτή η συνάρτηση λειτουργεί ως εξής, επιλέγει τα τελευταία order δείγματα από το σήμα. Η flip(...) αντιστρέφει τη σειρά των δειγμάτων, ώστε να τα χρησιμοποιεί με τη σωστή αντιστοιχία προς τους συντελεστές πρόβλεψης. Η coeffs .* flip(...) υπολογίζει το γινόμενο κάθε συντελεστή με το αντίστοιχο προηγούμενο δείγμα. Τέλος, αθροίζονται όλα τα παραπάνω γινόμενα για να δώσει την πρόβλεψη. Με το ίδιο σκεπτικό θα υλοποιήσουμε και μια ανώνυμη συνάρτηση για τον κβαντισμό μιας τιμής. Αυτή η συνάρτηση θα κβαντίζει μια τιμή σε προκαθορισμένα επίπεδα, περιορίζοντάς την στη δυναμική περιοχή. Οι παράμετροι της θα είναι οι:
value: Η τιμή που θα κβαντιστεί (π.χ., σφάλμα πρόβλεψης).
delta: Το βήμα του κβαντισμού (Δ).
min_value: Η ελάχιστη τιμή της δυναμικής περιοχής.
max_value: Η μέγιστη τιμή της δυναμικής περιοχής
Αρχικά διαιρεί την τιμή με το βήμα Δ, μετατρέποντας την σε "κλίμακα επιπέδων", μετά στρογγυλοποιεί στο πλησιέστερο επίπεδο κβαντισμού, επιστρέφει την κβαντισμένη τιμή στην αρχική δυναμική περιοχή πολλαπλασιάζοντας με Δ ενώ βεβαιώνεται ότι η τελική τιμή δεν ξεπερνά το max_value και δεν είναι μικρότερη από το min_value.

```
% Τάξη φίλτρου πρόβλεψης
predictor_order = length(predictor_coeffs);


%% Συνάρτηση για την πρόβλεψη
predict = @(coeffs, signal, n, order) ...
    sum(coeffs .* flip(signal(n - order + 1:n))); % Υπολογισμός πρόβλεψης

%% Συνάρτηση για κβαντισμό
quantize = @(value, delta, min_value, max_value) ...
    max(min(round(value / delta) * delta, max_value), min_value);
```

Πάμε να υλοποιήσουμε την κωδικοποίηση, ξεκινώντας ένα for loop που επαναλαμβάνεται για κάθε δείγμα του σήματος. Ο num_samples είναι το πλήθος των δειγμάτων του σήματος που επεξεργαζόμαστε. Ελέγχει πρώτα αν το τρέχον δείγμα είναι μέσα στην αρχική περιοχή (πριν από το πλήθος των προβλέψεων που απαιτούνται για την πρόβλεψη). Για τα πρώτα predictor_order δείγματα (δηλαδή για τα δείγματα που δεν έχουν προηγούμενα δείγματα για να κάνουμε πρόβλεψη), η πρόβλεψη ορίζεται στο 0. Δηλαδή, δεν κάνουμε καμία πρόβλεψη, και θεωρούμε το σφάλμα ίσο με το πραγματικό δείγμα. Για τα δείγματα που έχουν προηγούμενα δείγματα και μπορεί να υπολογιστεί η πρόβλεψη καλείται μια συνάρτηση predict, η οποία υπολογίζει την πρόβλεψη για το τρέχον δείγμα. Χρησιμοποιεί τα προηγούμενα δείγματα (reconstructed_signal) και τους συντελεστές πρόβλεψης (predictor_coeffs) για να υπολογίσει το προβλεπόμενο δείγμα. Το n - 1 δηλώνει το προηγούμενο δείγμα, και το predictor_order καθορίζει πόσα προηγούμενα δείγματα θα χρησιμοποιηθούν για τον υπολογισμό της πρόβλεψης. Μετά, υπολογίζουμε το σφάλμα πρόβλεψης, το οποίο είναι η διαφορά μεταξύ του πραγματικού δείγματος (t(n)) και του προβλεπόμενου δείγματος (prediction). Αυτό το σφάλμα είναι που χρησιμοποιούμε για την κβαντισμένη αναπαράσταση του σήματος.Στη συνέχεια καλείται η συνάρτηση quantize κβαντίζει το σφάλμα πρόβλεψης, το οποίο σημαίνει ότι το σφάλμα μετατρέπεται σε έναν από τους περιορισμένους αριθμούς που μπορούν να αναπαρασταθούν στον ψηφιακό κώδικα, με όρια κβαντοποίησης:
delta: Το μέγεθος του βήματος της κβαντοποίησης.
min_value και max_value: Τα όρια των τιμών του κβαντισμένου σφάλματος.
Αποθηκεύουμε το κβαντισμένο σφάλμα στο πίνακα quantized_signal. Αυτό είναι το τελικό σήμα που μεταδίδεται ή αποθηκεύεται. Τέλος, η ανακατασκευή του τρέχοντος δείγματος γίνεται προσθέτοντας το κβαντισμένο σφάλμα στην πρόβλεψη. Έτσι, το reconstructed_signal είναι το σήμα που ανακατασκευάζεται από την πρόβλεψη και το σφάλμα. Αυτό το σήμα χρησιμοποιείται για την εκτίμηση του σήματος στο επόμενο βήμα της διαδικασίας DPCM.

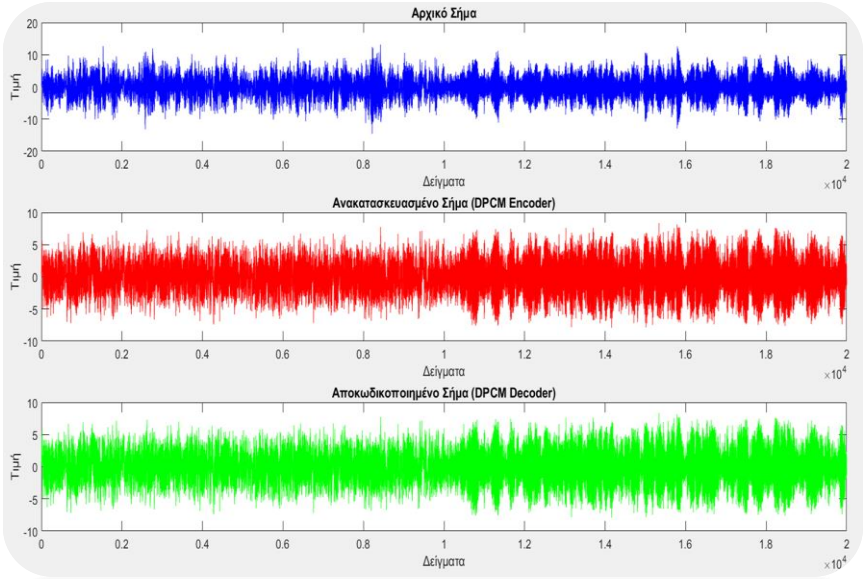
```
%% Κωδικοποίηση DPCM
for n = 1:num_samples
    if n <= predictor_order
        prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
    else
        % Υπολογισμός πρόβλεψης με βάση προηγούμενα δείγματα
        prediction = predict(predictor_coeffs, reconstructed_signal, n - 1,
predictor_order);
    end

    % Υπολογισμός σφάλματος πρόβλεψης
    error_signal(n) = t(n) - prediction;

    % Κβαντισμός σφάλματος
    quantized_error = quantize(error_signal(n), delta, min_value, max_value);

    % Αποθήκευση κβαντισμένου σφάλματος
    quantized_signal(n) = quantized_error;

    % Ανακατασκευή του τρέχοντος δείγματος
    reconstructed_signal(n) = prediction + quantized_error;
end
```

Τώρα υλοποιούμε την αποκωδικοποίηση, αρχικοποιώντας τον πίνακα decoded_signal διαστάσεων (num_samples x 1) με μηδενικά. Αυτός ο πίνακας θα αποθηκεύσει τα αποκωδικοποιημένα δείγματα του σήματος. Το num_samples αναφέρεται στον αριθμό δειγμάτων που θέλουμε να αποκωδικοποιήσουμε. Ξεκινάμε πάλι μια επανάληψη που επεξεργάζεται κάθε δείγμα του σήματος στην οποία ελέγχεται αν η τρέχουσα θέση του δείγματος, n, είναι μικρότερη ή ίση με τη σειρά πρόβλεψης (predictor_order). Ο predictor_order καθορίζει πόσα προηγούμενα δείγματα χρησιμοποιούνται για την πρόβλεψη του τρέχοντος δείγματος. Αν n <= predictor_order, δηλαδή αν βρισκόμαστε στα πρώτα δείγματα (όπου δεν υπάρχουν αρκετά προηγούμενα δείγματα για πρόβλεψη), η πρόβλεψη ορίζεται σε 0. Αυτό γίνεται επειδή δεν υπάρχει αρκετή πληροφορία για να υπολογιστεί μια πραγματική πρόβλεψη. Εάν n > predictor_order, τότε υπολογίζεται μια πρόβλεψη (prediction) για το τρέχον δείγμα, βασισμένη στα προηγούμενα αποκωδικοποιημένα δείγματα. Μετά καλείται η συνάρτηση predict, η οποία υπολογίζει την πρόβλεψη με βάση τα εξής:
predictor_coeffs: Οι συντελεστές του προγνωστικού φίλτρου που καθορίζουν πώς συνδυάζονται τα προηγούμενα δείγματα για την πρόβλεψη.
decoded_signal: Ο πίνακας με τα προηγούμενα αποκωδικοποιημένα δείγματα.

n - 1: Ο δείκτης του τελευταίου διαθέσιμου δείγματος.
predictor_order: Ο αριθμός των προηγούμενων δειγμάτων που χρησιμοποιούνται για την πρόβλεψη.
Η συνάρτηση predict επιστρέφει την πρόβλεψη για το τρέχον δείγμα. Τέλος, γίνεται η ανακατασκευή σήματος προσθέτοντας την πρόβλεψη (prediction) στο ποσοτικοποιημένο διαφορικό σήμα (quantized_signal(n)). Το quantized_signal(n) είναι η διαφορά μεταξύ του πραγματικού δείγματος και της πρόβλεψης, όπως είχε υπολογιστεί κατά την κωδικοποίηση και το αποτέλεσμα αποθηκεύεται στο decoded_signal(n). Με αυτόν τον τρόπο, ο κώδικας αποκωδικοποιεί το σήμα επαναφέροντάς το από τη συμπιεσμένη μορφή του.

```
%% Αποκωδικοποίηση DPCM
decoded_signal = zeros(num_samples, 1);
for n = 1:num_samples
    if n <= predictor_order
        prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
    else
        % Υπολογισμός πρόβλεψης με βάση τα προηγούμενα αποκωδικοποιημένα δείγματα
        prediction = predict(predictor_coeffs, decoded_signal, n - 1,
predictor_order);
    end

    % Ανακατασκευή δείγματος
    decoded_signal(n) = prediction + quantized_signal(n);
end
```

Το αποτέλεσμα που λαμβάνουμε είναι το εξής:



ΠΑΡΑΤΗΡΗΣΗ: Το διάγραμμα που προκύπτει από τον DPCM κωδικοποιητή μας δείχνει ότι το σήμα αυτό έχει μικρότερο εύρος τιμών σε σχέση με το αρχικό (μεταξύ -10 και +10). Αυτό είναι αναμενόμενο, καθώς η τεχνική DPCM αποθηκεύει μόνο τις διαφορές μεταξύ των διαδοχικών δειγμάτων, οι οποίες τείνουν να έχουν μικρότερη απόλυτη τιμή σε σχέση με τα αρχικά δείγματα. Η συμπίεση των τιμών είναι το κύριο πλεονέκτημα της DPCM, καθώς μειώνει το εύρος που χρειάζεται να αποθηκευτεί ή να μεταδοθεί, κάτι που εξοικονομεί χώρο ή εύρος ζώνης. Επίσης, το διάγραμμα που είναι το αποτέλεσμα της αποκωδικοποίησης του DPCM κωδικοποιημένου σήματος, μας δείχνει ότι το σήμα είναι αρκετά παρόμοιο με το αρχικό μπλε διάγραμμα, γεγονός που υποδεικνύει ότι η διαδικασία αποκωδικοποίησης λειτουργεί σωστά. Οι μικρές αποκλίσεις που παρατηρούνται, κυρίως σε ορισμένες κορυφές, είναι τυπικές για την τεχνική DPCM και προκύπτουν από απώλειες λόγω ποσοτικοποίησης (η διαφορά μεταξύ του πραγματικού σήματος και της πρόβλεψης στρογγυλοποιείται στο πλησιέστερο επίπεδο ποσοτικοποίησης) ή από διάφορα μικρά λάθη πρόβλεψης.

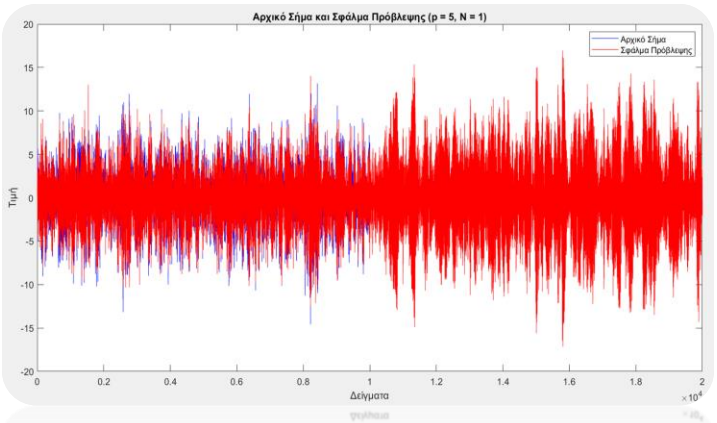
(ii). Τώρα πρέπει να επιλέξουμε δύο τιμές του $p \geq 5$ και για N = 1,2,3 bits και να συγκρίνουμε το αρχικό σήμα και το σφάλμα πρόβλεψης y. Φορτώνουμε τα δεδομένα από το αρχείο source.mat, ορίζουμε ξανά τη δυναμική περιοχή των τιμών του σήματος που θα ποσοτικοποιηθεί (οι τιμές min_value καιmax_value θα χρησιμοποιηθούν για τον υπολογισμό των βημάτων κβαντιστή (quantization step size) και ορίζουμε τις τιμές του p, που είναι ο αριθμός των προηγούμενων δειγμάτων που θα χρησιμοποιηθούν για τη δημιουργία του προγνωστικού φίλτρου. Επιλέγουμε τις τιμές 5 και 7. Έπειτα, υπολογίζουμε τον αριθμό των δειγμάτων του σήματος t, το αποθηκεύουμε στη μεταβλητή num_samples και ξεκινάει ο βρόχος όπου επεξεργάζεται το σήμα για κάθε τιμή του p που έχει οριστεί στο p_values ([5, 7]). Το predictor_coeffs είναι ένας πίνακας με p συντελεστές, όλοι είναι ίσοι με 1/p. Αυτό το φίλτρο υπολογίζει τον απλό μέσο όρο των τελευταίων p δειγμάτων. Ο predictor_order αποθηκεύει το μέγεθος του φίλτρου, δηλαδή τον αριθμό των συντελεστών. Μέσα σε αυτό τον βρόχο ξεκινάει ένας εσωτερικός βρόχος για τρεις διαφορετικές τιμές του N (1, 2, 3). Το N αντιπροσωπεύει τον αριθμό των bits που θα χρησιμοποιηθούν για την ποσοτικοποίηση. Πρώτα υπολογίζει τον αριθμό των επιπέδων του κβαντιστή με βάση τον αριθμό των bits και μετά υπολογίζει το βήμα του κβαντιστή (quantization step size), δηλαδή τη διαφορά μεταξύ δύο διαδοχικών επιπέδων.

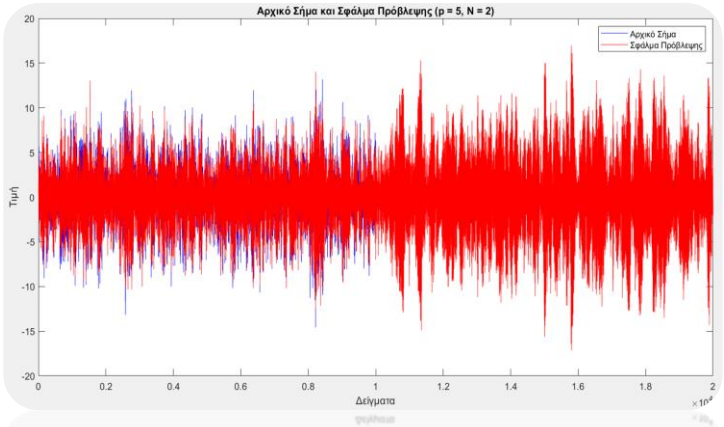
```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής

% Επιλογή τιμών του p
p_values = [5, 7]; % Δύο διαφορετικές τιμές του p

% Αρχικοποίηση
num_samples = length(t);

% Σχεδίαση για κάθε τιμή του p
for p = p_values
    % Δημιουργία φίλτρου πρόβλεψης με p συντελεστές
    predictor_coeffs = ones(1, p) / p; % Απλό μέσο όρο p προηγούμενων τιμών
    predictor_order = length(predictor_coeffs);

    for N = 1:3 % Για κάθε αριθμό bits
        % Υπολογισμός παραμέτρων κβαντιστή
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή
```

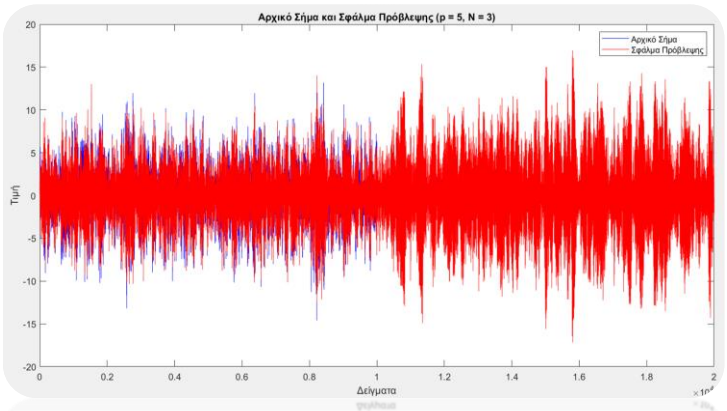
Πάμε να υπολογίσουμε το σήμα πρόβλεψης, το σφάλμα πρόβλεψης και το κβαντισμένο σφάλμα για το σήμα. Δημιουργούμε τρεις μηδενικούς πίνακες, τον quantized_signal με μήκος ίσο με το σήμα t (θα αποθηκεύει το κβαντισμένο σφάλμα πρόβλεψης για κάθε δείγμα), τον reconstructed_signal για την αποθήκευση του ανακατασκευασμένου σήματος (που θα προκύψει από τον αποκωδικοποιητή) και τον error_signal για την αποθήκευση του σφάλματος πρόβλεψης σε κάθε δείγμα. Ξεκινάει ένας βρόχος που επεξεργάζεται κάθε δείγμα t(n) από το σήμα εισόδου, με n να κυμαίνεται από 1 έως num_samples. Πάλι αν δεν υπάρχουν αρκετά προηγούμενα δείγματα για να γίνει πρόβλεψη, οπότε η πρόβλεψη prediction ορίζεται σε 0. Αλλιώς, υπολογίζεται η πρόβλεψη χρησιμοποιώντας τα predictor_order προηγούμενα δείγματα. Η t(n - predictor_order:n - 1)' παίρνει έναν πίνακα με τα predictor_order προηγούμενα δείγματα του σήματος t. Η predictor_coeffs .* t(...) υπολογίζει το γινόμενο κάθε συντελεστή πρόβλεψης (predictor_coeffs) με το αντίστοιχο προηγούμενο δείγμα και αθροίζονται τα γινόμενα για να υπολογίσει την πρόβλεψη prediction. Υπολογίζεται το σφάλμα πρόβλεψης (διαφορά μεταξύ του τρέχοντος δείγματος t(n) και της πρόβλεψης prediction) και αποθηκεύεται στον πίνακα error_signal στη θέση n. Το σφάλμα πρόβλεψης error_signal(n) κβαντίζεται ώστε να περιοριστεί η δυναμική του και να μετατραπεί σε διακριτές τιμές. Διαιρούμε το σφάλμα με το βήμα κβαντιστή delta, μετατρέποντας το σε κλίμακα επιπέδων κβαντιστή, στρογγυλοποιούμε το αποτέλεσμα στον πλησιέστερο ακέραιο (δηλαδή, στο πλησιέστερο επίπεδο κβαντιστή) και πολλαπλασιάζουμε με το delta για να επιστρέψει το κβαντισμένο σφάλμα στη δυναμική περιοχή του σήματος. Περιορίζουμε το κβαντισμένο σφάλμα στα όρια min_value και max_value, διασφαλίζοντας ότι δεν θα ξεφύγει από τη δυναμική περιοχή. Το κβαντισμένο σφάλμα quantized_error αποθηκεύεται στον πίνακα quantized_signal στη θέση n.

```
% Αρχικοποίηση σημάτων
        quantized_signal = zeros(num_samples, 1);
        reconstructed_signal = zeros(num_samples, 1);
        error_signal = zeros(num_samples, 1);

        % Υπολογισμός σφάλματος πρόβλεψης
        for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης (χρήση προηγούμενων δειγμάτων)
                prediction = sum(predictor_coeffs .* t(n - predictor_order:n - 1)');
            end

            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal(n) = t(n) - prediction;

            % Κβαντισμός σφάλματος
            quantized_error = max(min(round(error_signal(n) / delta) * delta,
max_value), min_value);
            quantized_signal(n) = quantized_error;
        end
```

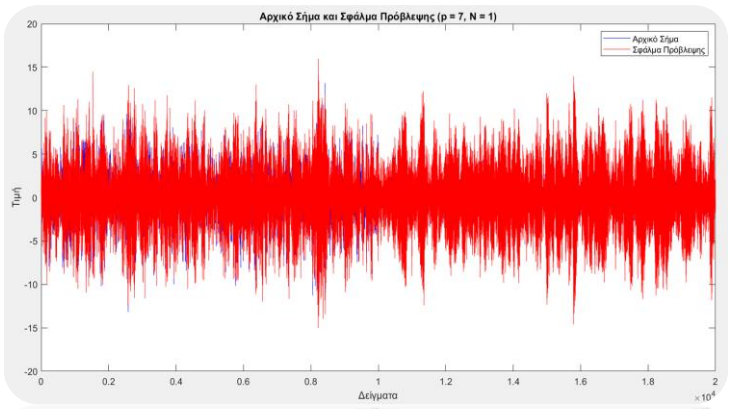
Τα αποτελέσματα μας θα είναι τα ακόλουθα: Για p = 5:

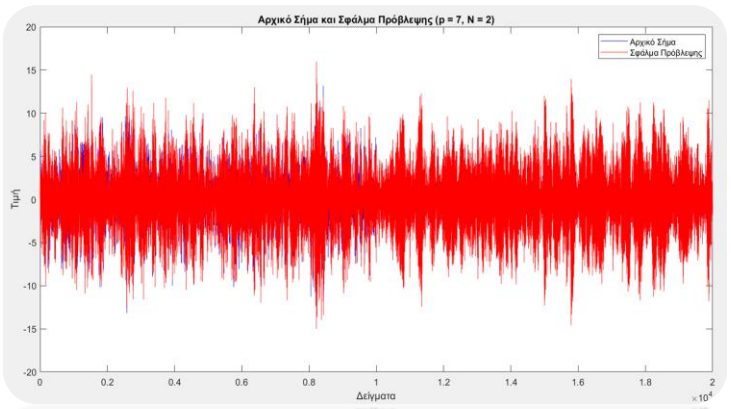


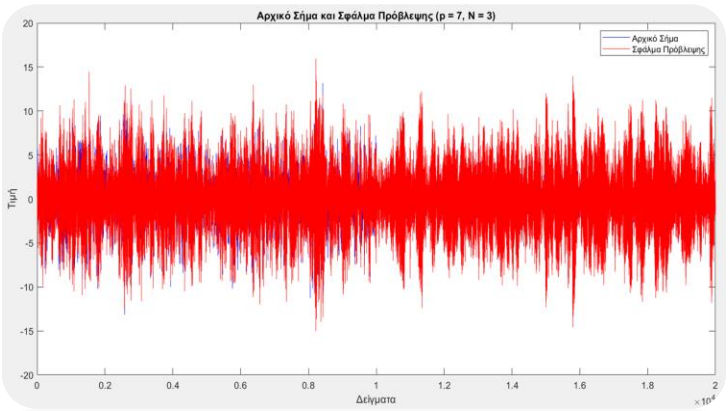




Για p = 7:







<u>ΠΑΡΑΤΗΡΗΣΗ:</u> Για p=5:
Το σφάλμα πρόβλεψης (κόκκινη γραμμή) φαίνεται να ακολουθεί το πρωτότυπο σήμα (μπλε γραμμή) αλλά με μεγαλύτερη ακρίβεια για μεγαλύτερο N (πλήθος bits). Η πρόβλεψη εξαρτάται από τον αριθμό των προηγούμενων δειγμάτων που λαμβάνονται υπόψη. Με N=1 bit, το επίπεδο ποσοτικοποίησης είναι περιορισμένο, και αυτό δημιουργεί αρκετό σφάλμα. Στο γράφημα, το κόκκινο σφάλμα παρουσιάζει αρκετή απόκλιση σε σύγκριση με το αρχικό σήμα.Η ποιότητα της πρόβλεψης είναι χαμηλότερη καθώς η ακρίβεια είναι περιορισμένη. Με N =2,3 bits, η ακρίβεια αυξάνεται λόγω των περισσότερων επιπέδων ποσοτικοποίησης. Παρατηρούμε ότι το σφάλμα είναι πιο κοντά στο αρχικό σήμα. Αυτό φαίνεται να είναι μια πιο αποδεκτή προσέγγιση, καθώς μειώνεται η απώλεια ακρίβειας. Το p=5 προσφέρει έναν μέσο όρο πρόβλεψης, λαμβάνοντας υπόψη 5 προηγούμενα δείγματα. Η επιλογή είναι ικανοποιητική για μέτρια ακρίβεια πρόβλεψης, αλλά με μεγαλύτερο N (bits), το αποτέλεσμα θα βελτιωνόταν περαιτέρω.
Για p=10:
Το αρχικό σήμα (μπλε) και το σφάλμα πρόβλεψης (κόκκινο) με N=1 bit φαίνονται να έχουν σχετικά παρόμοια χαρακτηριστικά. Ωστόσο, το σφάλμα πρόβλεψης είναι πιο διακυμαντικό και φαίνεται να περιέχει θόρυβο. Το γεγονός ότι το σφάλμα παραμένει κοντά στο 0 υποδεικνύει ότι το μοντέλο μπορεί να προβλέψει με κάποια ακρίβεια το αρχικό σήμα, αλλά υπάρχουν περιθώρια βελτίωσης.Με N=2, με την αύξηση της παραμέτρου N, το σφάλμα πρόβλεψης εξακολουθεί να εμφανίζει σημαντικές διακυμάνσεις, αλλά φαίνεται να μειώνεται η συχνότητα με την οποία αποκλίνει έντονα από το 0. Αυτό δείχνει ότι το μοντέλο ενδεχομένως μαθαίνει καλύτερα και το σφάλμα πρόβλεψης βελτιώνεται ελαφρώς Με N=3, παρατηρούμε ακόμα πιο λεία κατανομή του σφάλματος πρόβλεψης. Οι αποκλίσεις από το 0 είναι λιγότερο συχνές και λιγότερο έντονες. Αυτό υποδηλώνει ότι η ακρίβεια του μοντέλου βελτιώνεται περαιτέρω με την αύξηση του N, καθώς φαίνεται να προσαρμόζεται καλύτερα στο αρχικό σήμα. Όπως φαίνεται, καθώς αυξάνεται το N, το σφάλμα πρόβλεψης μειώνεται και η πρόβλεψη του σήματος γίνεται πιο ακριβής. Το p=7 οδηγεί στο να εξετάζονται περισσότερες προηγούμενες τιμές, κάτι που μπορεί να βελτιώσει την απόδοση σε σήματα με μακροχρόνιες εξαρτήσεις (long memory).

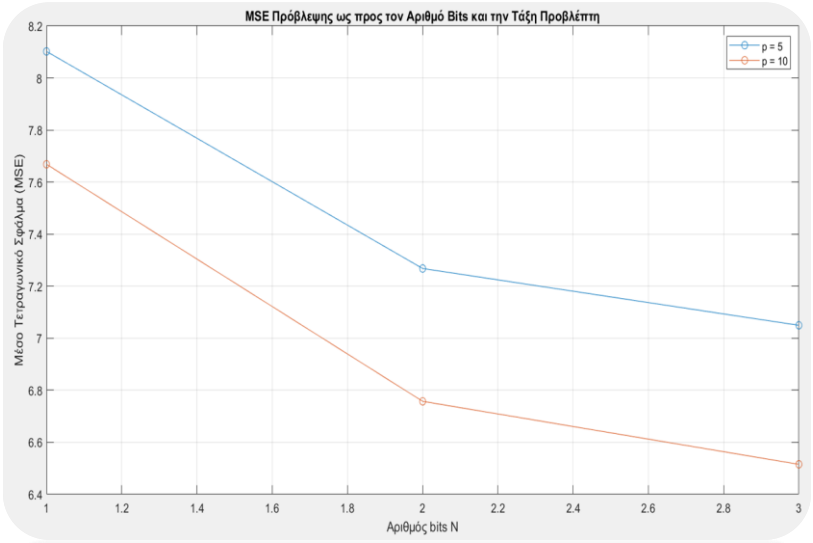
(iii). Δημιουργούμε έναν πίνακα mse_results όπου θα αποθηκευτούν τα αποτελέσματα MSE για κάθε συνδυασμό τιμών p και N. Ο πίνακας έχει διαστάσεις length(p_values)×length(N_values), δηλαδή έναν αριθμό γραμμών ίσο με τις διαφορετικές τιμές του p και έναν αριθμό στηλών ίσο με τις διαφορετικές τιμές του N. Ο εξωτερικός βρόχος που ακολουθεί περνά από κάθε τιμή του p στη λίστα p_values. Εκεί δημιουργείται ένας πίνακας συντελεστών predictor_coeffs, που αντιστοιχεί σε έναν προβλέπτη που υπολογίζει το μέσο όρο των p προηγούμενων τιμών. Οι συντελεστές είναι όλοι ίσοι με 1/p, καθώς πρόκειται για ομοιόμορφα κατανεμημένους συντελεστές. Η μεταβλητή predictor_orderpredictor_order αποθηκεύει το πλήθος των συντελεστών. Χρησιμοποιούμε τη fprintf για να εκτυπώσουμε τους συντελεστές προβλέπτη. Ο εσωτερικός βρόχος επαναλαμβάνεται για κάθε τιμή του N στη λίστα N_values. Ο αριθμός επιπέδων του κβαντιστή διπλασιάζεται για κάθε επιπλέον bit. Το βήμα του κβαντιστή (Δ) υπολογίζεται διαιρώντας τη δυναμική περιοχή του σήματος από min_value έως max_value με το πλήθος των επιπέδων κβαντιστή. Συνεχίζουμε υπολογίζοντας τον αριθμό των δειγμάτων του σήματος t και αρχικοποιούμε το σφάλμα δημιουργώντας έναν πίνακα error_signal, όπου θα αποθηκεύεται το σφάλμα πρόβλεψης για κάθε δείγμα του σήματος.

```
% Αρχικοποίηση αποτελεσμάτων
mse_results = zeros(length(p_values), length(N_values)); % Μέσο τετραγωνικό σφάλμα για
κάθε συνδυασμό

% Υπολογισμός MSE για κάθε συνδυασμό p και N
for p_idx = 1:length(p_values)
    p = p_values(p_idx); % Τρέχουσα τιμή του p

    % Δημιουργία συντελεστών προβλέπτη (απλός μέσος όρος p τιμών)
    predictor_coeffs = ones(1, p) / p; % Ομοιόμορφα κατανεμημένοι συντελεστές
    predictor_order = length(predictor_coeffs);
    fprintf('Συντελεστές προβλέπτη για p = %d: %s\n', p, mat2str(predictor_coeffs));

    for N_idx = 1:length(N_values)
        N = N_values(N_idx); % Τρέχουσα τιμή των bits
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

        % Υπολογισμός σφάλματος πρόβλεψης
        num_samples = length(t);
        error_signal = zeros(num_samples, 1);
```

Υπολογίζουμε το σφάλμα πρόβλεψης όπως και στο παραπάνω ερώτημα και υπολογίζουμε το μέσο τετραγωνικό σφάλμα (MSE) για τον συγκεκριμένο συνδυασμό p και N. Ο πίνακας mse_resultsmse_results ενημερώνεται με το αποτέλεσμα στη θέση (p_idx,N_idx).

```
for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης
                prediction = sum(predictor_coeffs .* t(n - predictor_order:n - 1)');
            end

            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal(n) = t(n) - prediction;

            % Κβαντισμός σφάλματος
            error_signal(n) = max(min(round(error_signal(n) / delta) * delta,
max_value), min_value);
        end

        % Υπολογισμός MSE
        mse_results(p_idx, N_idx) = mean(error_signal.^2);
    end
end
```

Παρακάτω φαίνεται το γράφημα που να δείχνει το μέσο τετραγωνικό σφάλμα πρόβλεψης ως προς το N για p=5,10:



ΠΑΡΑΤΗΡΗΣΗ: Παρατηρούμε ότι και για τις δύο τιμές του p, το MSE μειώνεται καθώς αυξάνονται τα bits N. Αυτό είναι αναμενόμενο, καθώς περισσότερα bits σημαίνουν μεγαλύτερη ακρίβεια στην αναπαράσταση δεδομένων, που οδηγεί σε καλύτερη πρόβλεψη. Για p=10, το MSE είναι γενικά χαμηλότερο σε σύγκριση με p=5. Αυτό δείχνει ότι η μεγαλύτερη τάξη προβλέπτη (μεγαλύτερη πολυπλοκότητα) βελτιώνει την απόδοση, καθώς χρησιμοποιούνται περισσότερες πληροφορίες για την πρόβλεψη. Επίσης, όταν ο αριθμός των bits N είναι μικρός (π.χ. N=1), η διαφορά μεταξύ p=5 και p=10 είναι πιο εμφανής. Καθώς το N αυξάνεται (π.χ. N=3), οι καμπύλες πλησιάζουν, κάτι που δείχνει ότι η αύξηση των bits μπορεί να εξαλείψει σε μεγάλο βαθμό τα οφέλη μιας υψηλότερης τάξης p.

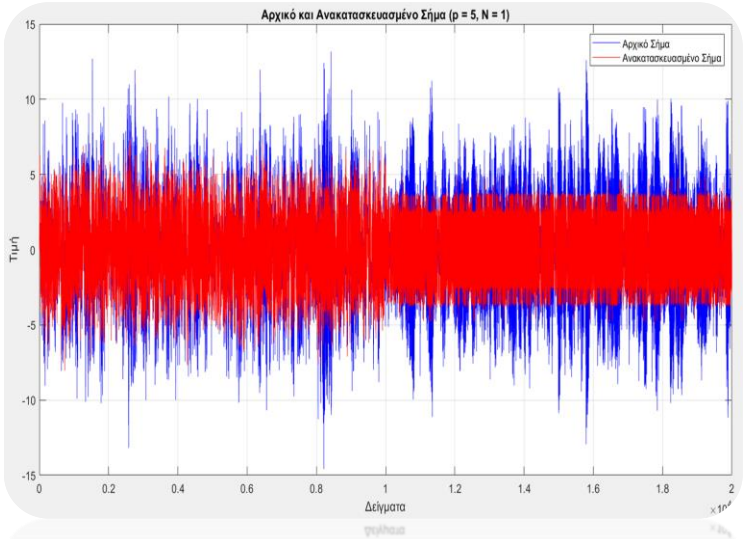
(iv). Σε αυτό το ερώτημα επεκτείνεται η λειτουργικότητα του προηγούμενου τμήματος και προστίθεται η ανακατασκευή του σήματος χρησιμοποιώντας την πρόβλεψη και τον κβαντισμό. Εδώ υπολογίζεται το ανακατασκευασμένο σήμα και αποθηκεύεται το αποτέλεσμα για κάθε συνδυασμό p και N. Ο εξωτερικός βρόχος επαναλαμβάνεται για κάθε τιμή του p στη λίστα p_values δημιουργώντας ένα προβλέπτη βασισμένο στον μέσο όρο p τιμών με όλους τους συντελεστές του προβλέπτη να είναι 1/p και το πλήθος των συντελεστών να είναι ίσο με p. Κατά τα γνωστά, ο εσωτερικός βρόχος που ακολουθεί επαναλαμβάνεται για κάθε τιμή του N στη λίστα N_values και υπολογίζει τον αριθμό επιπέδων κβαντιστή, που είναι 2^N και τα βήμα κβαντιστή(Δ), δηλαδή το εύρος κάθε επιπέδου ενώ παράλληλα δημιουργούνται δύο πίνακες, ένας για αποθήκευση του κβαντισμένου σφάλματος πρόβλεψης για κάθε δείγμα και ένας για αποθήκευση του ανακατασκευασμένου σήματος για κάθε δείγμα. Μετά για κάθε δείγμα του σήματος t(n) προβλέπουμε για τα πρώτα δείγματα όπου πάλι εάν το τρέχον δείγμα n είναι μικρότερο ή ίσο με το πλήθος των συντελεστών του προβλέπτη (predictor_order=p), δεν υπάρχουν αρκετά προηγούμενα δείγματα και η πρόβλεψη prediction ορίζεται ως 0. Εάν n>p, η πρόβλεψη prediction υπολογίζεται ως το εσωτερικό γινόμενο (dot product) των συντελεστών του προβλέπτη predictor_coeffs με τα p πιο πρόσφατα δείγματα του ανακατασκευασμένου σήματος reconstructed_signal. Αντί να χρησιμοποιούνται τα πραγματικά δεδομένα του t, χρησιμοποιείται το ανακατασκευασμένο σήμα reconstructed_signal. Αυτό διασφαλίζει ότι ο υπολογισμός βασίζεται μόνο σε δεδομένα που ήδη έχουν προβλεφθεί και κβαντιστεί. Έπειτα το σφάλμα πρόβλεψης error_signal υπολογίζεται όπως ήδη γνωρίζουμε ως η διαφορά του πραγματικού δείγματος t(n) από την πρόβλεψη prediction. Τέλος, το error_signal κβαντίζεται στα κοντινότερα επίπεδα του κβαντιστή με το σφάλμα να διαιρείται με το βήμα Δ και στρογγυλοποιείται στην πλησιέστερη ακέραια τιμή, ενώ μετά πολλαπλασιάζεται ξανά με το Δ για να επιστρέψει στο αρχικό εύρος

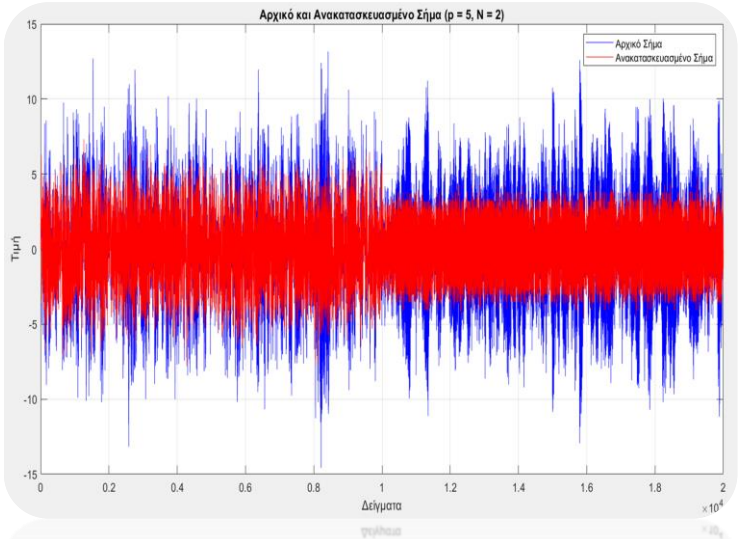
λειτουργώντας στη δυναμική περιοχή [min_value,max_value]. Το κβαντισμένο σφάλμα quantized_error αποθηκεύεται στον πίνακα quantized_signal. Το ανακατασκευασμένο σήμα reconstructed_signal(n)

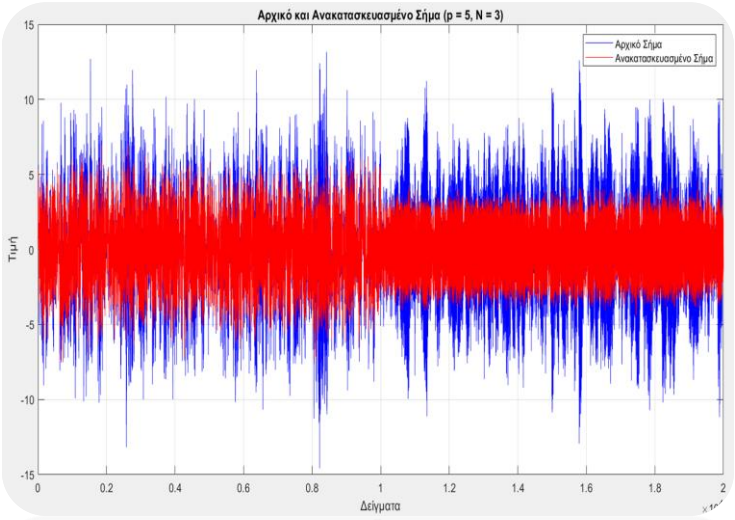
υπολογίζεται προσθέτοντας την πρόβλεψη prediction και το κβαντισμένο σφάλμα quantized_error. Εδώ η ανακατασκευή χρησιμοποιεί το σφάλμα που έχει ήδη κβαντιστεί, ώστε να προσομοιώνει τη διαδικασία αποκωδικοποίησης.

```
% Υπολογισμός και σχεδίαση για κάθε συνδυασμό p και N
for p_idx = 1:length(p_values)
    p = p_values(p_idx); % Τρέχουσα τιμή του p

    % Δημιουργία συντελεστών προβλέπτη (απλός μέσος όρος p τιμών)
    predictor_coeffs = ones(1, p) / p; % Ομοιόμορφα κατανεμημένοι συντελεστές
    predictor_order = length(predictor_coeffs);

    for N_idx = 1:length(N_values)
        N = N_values(N_idx); % Τρέχουσα τιμή των bits
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

        % Υπολογισμός ανακατασκευασμένου σήματος
        quantized_signal = zeros(num_samples, 1);
        reconstructed_signal = zeros(num_samples, 1);

        for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης
                prediction = sum(predictor_coeffs .* reconstructed_signal(n -
predictor_order:n - 1)');
            end

            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal = t(n) - prediction;

            % Κβαντισμός σφάλματος
            quantized_error = max(min(round(error_signal / delta) * delta,
max_value), min_value);
            quantized_signal(n) = quantized_error;

            % Ανακατασκευή σήματος
            reconstructed_signal(n) = prediction + quantized_error;
        end
```

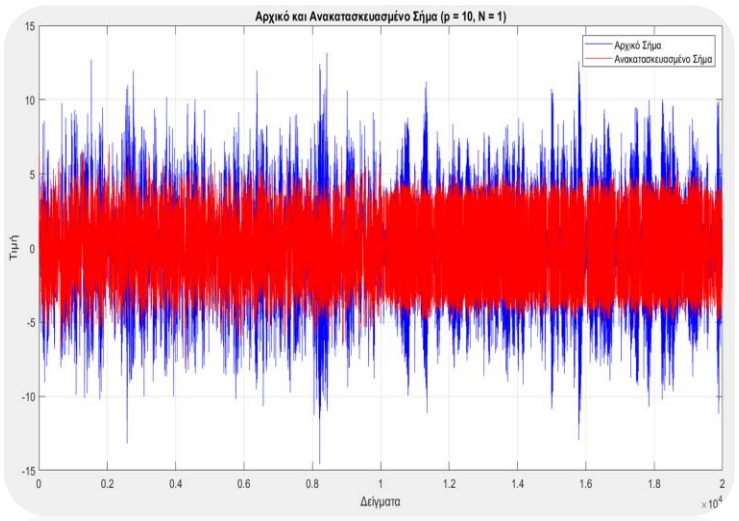
Παρακάτω βλέπουμε το αρχικό και το ανακατασκευασμένο σήμα στο ίδιο γράφημα έχοντας επιλέξει για p = 5:

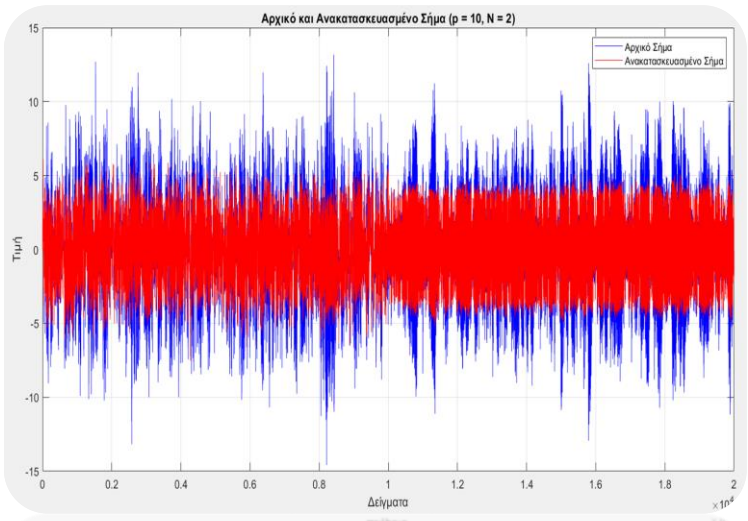


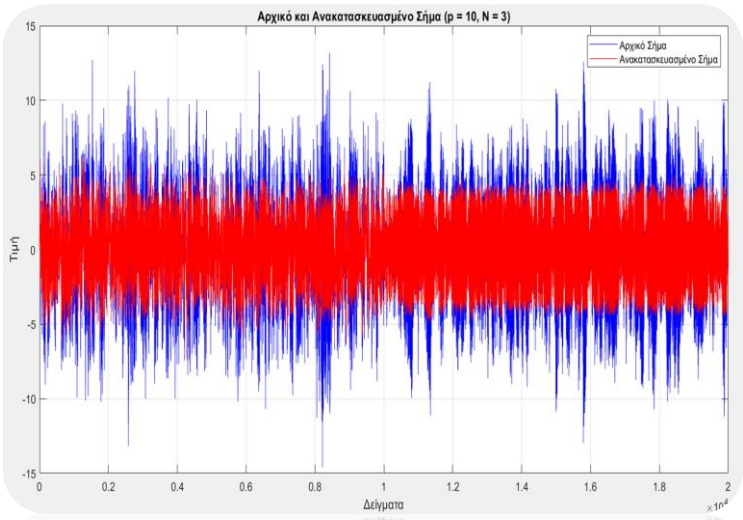




Παρακάτω βλέπουμε το αρχικό και το ανακατασκευασμένο σήμα στο ίδιο γράφημα έχοντας επιλέξει για p = 10:







ΠΑΡΑΤΗΡΗΣΗ: Παρατηρούμε ότι όντως η καλύτερη ανακατασκευή πραγματοποιείται όταν είναι p = 10 και N = 3, ενώ η χειρότερη όταν είναι p = 5 και N = 1.
Γενικά, η αποδοτικότερη ανακατασκευή συμβαίνει με τον ιδανικό συνδυασμό μιας μεγάλης τιμής N και μιας μεγάλης τιμής p. Πράγματι βλέπουμε ότι η ανακατασκευή που προκύπτει για p = 10 αλλά N = 1 απέχει πάρα πολύ από το πραγματικό σήμα παρά τη μεγάλη τιμή p, όπως επίσης το ίδιο συμβαίνει και για p = 5 αλλά N = 3 παρά τη μεγάλη τιμή του N. Επίσης, παρατηρούμε ότι η ανακατασκευή σε συγκεκριμένα σημεία είναι ακριβέστερη από ότι στο συνολικό σήμα.

ΜΕΡΟΣ Β: Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος M-PAM

(i). Με βάση τις παραπάνω υποδείξεις, υλοποιήστε το σύστημα M-PAM και αναφερθείτε στα βασικά του σημεία.
(ii). Μετρήστε την πιθανότητα σφάλματος bit και σχεδιάστε την καμπύλη BER για M = 2 & 8 για απλή κωδικοποίηση για τιμές του SNR = 0: 2: 20 dB. Επαναλάβετε το ερώτημα για M = 8 αν τα σύμβολα στην αντιστοίχιση κωδικοποιούνταν κατά Gray. Οι καμπύλες να σχεδιαστούν όλες στο ίδιο γράφημα.
(iii). Μετρήστε την πιθανότητα σφάλματος συμβόλου και σχεδιάστε την καμπύλη SER για M = 2,8 για απλή κωδικοποίηση για τιμές του SNR = 0: 2: 20 dB. Οι καμπύλες θα πρέπει και πάλι να σχεδιαστούν όλες στο ίδιο γράφημα.

(i). Το σύστημα M-PAM (M-level Pulse Amplitude Modulation) είναι μια τεχνική διαμόρφωσης σήματος στην οποία χρησιμοποιούνται M διαφορετικά επίπεδα πλάτους (amplitude levels) για να εκπροσωπήσουν δεδομένα. Είναι μια μορφή πολικής διαμόρφωσης πλάτους που συνήθως χρησιμοποιείται σε ψηφιακά συστήματα επικοινωνίας για την αποστολή δεδομένων μέσω αναλογικών σημάτων. Αν το M είναι 4 (δηλαδή 4-PAM), τότε το σήμα θα έχει 4 διαφορετικά επίπεδα πλάτους, το οποίο αντιστοιχεί σε 2 bits ανά σύμβολο (δηλαδή 00, 01, 10, 11). Πάμε να υλοποιήσουμε το σύστημα M-PAM για 8 σύμβολα. Ξεκινάμε ορίζοντας τον αριθμό των συμβόλων που θα μεταδοθούν, την συχνότητα φέρουσας (σε Hz) για την αναλογική διαμόρφωση του σήματος, την συχνότητα δειγματοληψίας (σε Hz) για το σήμα, την διάρκεια του συμβόλου (σε δευτερόλεπτα) και τον χρονικό άξονα για τη διάρκεια κάθε συμβόλου (με δείγματά σε συχνότητα fs). Ο αστερισμός A = 2 * (1:M) - M - 1 δημιουργεί τα επίπεδα πλάτους για την διαμόρφωση 8-PAM. Στην περίπτωση του 8-PAM, τα επίπεδα πλάτους είναι [−4,−2,0,+2, +4].Μετά δημιουργείται μια τυχαία ακολουθία από σύμβολα, όπου κάθε σύμβολο επιλέγεται τυχαία από τα επίπεδα πλάτους A. Η συνάρτηση randi δημιουργεί μια τυχαία ακολουθία από σύμβολα, όπου κάθε σύμβολο επιλέγεται τυχαία από τα επίπεδα πλάτους A.

```
M = 8; % Αριθμός συμβόλων (M-PAM)
N = 1000; % Αριθμός συμβόλων που θα σταλούν
fc = 1000; % Συχνότητα φέρουσας (Hz)
fs = 10000; % Συχνότητα δειγματοληψίας (Hz)
T = 1/100; % Διάρκεια συμβόλου (s)
t = 0:1/fs:T-1/fs; % Χρονικός άξονας για κάθε σύμβολο

% Αστερισμός M-PAM
A = 2 * (1:M) - M - 1;

% Τυχαία συμβολική ακολουθία
symbols = A(randi([1, M], 1, N));
```

Η s_t = zeros(1, length(t) * N) δημιουργεί το σήμα που θα μεταδοθεί, το οποίο θα έχει N σύμβολα, καθένα από τα οποία θα έχει διάρκεια T. Ακολουθεί ο for βρόχος που περνά από κάθε σύμβολο και το διαμορφώνει με τη φέρουσα (cos(2*pi*fc*t)), πολλαπλασιάζοντας κάθε σύμβολο με την αντίστοιχη φέρουσα και τοποθετώντας το στο σήμα εξόδου s_t ώστε να παράγεται το αντίστοιχο αναλογικό σήμα. Κάθε σύμβολο από την ακολουθία των δεδομένων θα διαμορφώνεται με τη φέρουσα. Η διαδικασία προσθήκης θορύβου είναι η εξής:
Ορίζεται το SNR (Σήμα προς Θόρυβο) σε dB με τη μεταβλητή SNR. Η τιμή αυτή καθορίζει πόσο ισχυρό είναι το σήμα σε σχέση με το θόρυβο.
Υπολογίζεται η ισχύς του σήματος ως το μέσο τετράγωνο της τιμής του σήματος από τον τύπο:

$$\text{signal\_power} = \text{mean}(st\ ^2)$$

Η ισχύς του θορύβου υπολογίζεται από το SNR:

$$\text{noise\_power} = \frac{\text{signal\_power}}{10^{\frac{SNR}{10}}}$$

Αυτό εξασφαλίζει ότι η ισχύς του θορύβου είναι σωστά κλιμακωμένη με το SNR. Ο θόρυβος δημιουργείται με τη χρήση της συνάρτησης randn, η οποία παράγει τυχαίες τιμές από μια κανονική κατανομή (Gaussian noise). Ο θόρυβος είναι κλιμακωμένος έτσι ώστε η ισχύς του να είναι ανάλογη με το noise_power. Το θορυβώδες σήμα r_t προκύπτει προσθέτοντας τον θόρυβο στο διαμορφωμένο σήμα rt=st+noise.Το r_t είναι το σήμα που περνάει από το κανάλι, το οποίο έχει υποστεί θόρυβο. Η αποδιαμόρφωση είναι η διαδικασία μέσω της οποίας ο δέκτης ανακτά τα δεδομένα από το θορυβώδες σήμα που έλαβε. Ακολουθεί και η διαδικασία της αποδιαμόρφωσης, δηλαδή το θορυβώδες σήμα r_t περνά από τον δέκτη, ο οποίος αποδιαμορφώνει το σήμα, προσπαθώντας να ανακτήσει τα αρχικά σύμβολα. Ο δέκτης προσπαθεί να ανακτήσει το επίπεδο πλάτους κάθε σύμβολο. Αυτό σημαίνει ότι το σήμα αναλύεται για κάθε σύμβολο, αναζητώντας την αντίστοιχη φάση της φέρουσας (το cos(2*pi*fc*t)). Η συνάρτηση:

$$r(i) = \frac{\sum(segment * \cos(2 * pi * fc * t))}{lenaht(t)}$$

υπολογίζει το εσωτερικό γινόμενο του σήματος με την φέρουσα, το οποίο χρησιμοποιείται για την εκτίμηση του πλάτους του σύμβολου για το i-οστό σύμβολο. Αυτή η τιμή r(i) αντιπροσωπεύει την εκτίμηση του επιπέδου πλάτους του μεταδιδόμενου σήματος (μεθοδολογία πλάτους μέγιστης πιθανότητας).

```
% Διαμόρφωση
s_t = zeros(1, length(t) * N);
for i = 1:N
    s_t((i-1)*length(t)+1:i*length(t)) = symbols(i) * cos(2*pi*fc*t);
end

% Προσθήκη θορύβου (χειροκίνητα)
SNR = 20; % Σήμα προς Θόρυβο σε dB
signal_power = mean(s_t.^2); % Υπολογισμός ισχύος σήματος
noise_power = signal_power / (10^(SNR/10)); % Υπολογισμός ισχύος θορύβου
noise = sqrt(noise_power) * randn(size(s_t)); % Δημιουργία Gaussian θορύβου
r_t = s_t + noise; % Προσθήκη θορύβου στο σήμα

% Αποδιαμόρφωση
r = zeros(1, N);
for i = 1:N
    segment = r_t((i-1)*length(t)+1:i*length(t));
    r(i) = sum(segment .* cos(2*pi*fc*t)) / length(t);
end
```

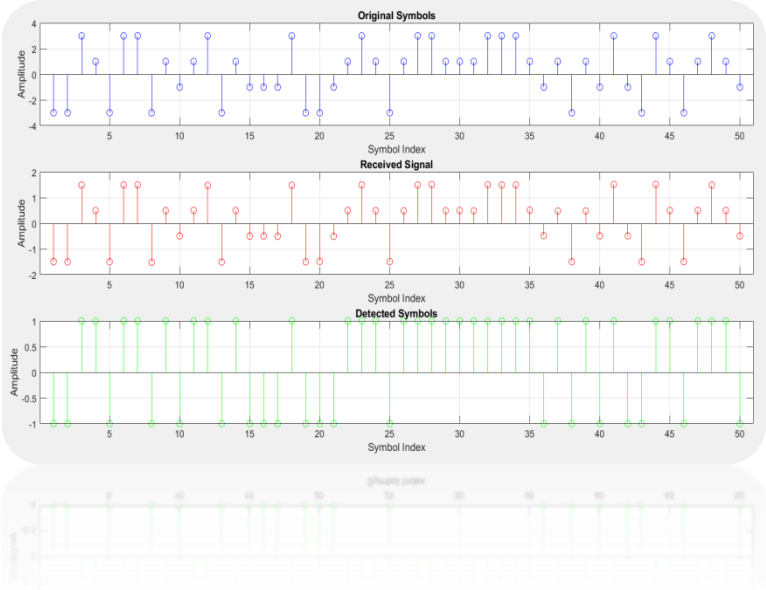
Μετά την αποδιαμόρφωση του σήματος, το επόμενο βήμα είναι η αναγνώριση των συμβόλων που στάλθηκαν από τον πομπό. Αυτό γίνεται με την εκτίμηση του πιο κοντινού επιπέδου πλάτους από το σύνολο των επιπέδων του αστερισμού (constellation). Για κάθε αποδιαμορφωμένο σύμβολο r(i), υπολογίζεται ποιο από τα επίπεδα πλάτους στο σύνολο A είναι το πιο κοντινό. Αυτό γίνεται με την συνάρτηση [,idx]=min(|A−r(i)|). Εδώ, το idx είναι ο δείκτης του επιπέδου πλάτους του αστερισμού που είναι το πιο κοντινό στο αποδιαμορφωμένο σύμβολο r(i). Η συνάρτηση min βρίσκει την ελάχιστη απόσταση μεταξύ του αποδιαμορφωμένου σήματος και των επιπέδων πλάτους A. Στη συνέχεια, το σύμβολο που αντιστοιχεί σε αυτό το επίπεδο πλάτους αποθηκεύεται στο διάνυσμα detected_symbols(i)=A(idx), δηλαδή, το detected_symbols(i) παίρνει την τιμή του επιπέδου πλάτους του αστερισμού που έχει το μικρότερο διαφορά από το αποδιαμορφωμένο σήμα.
Αφού γίνει η αναγνώριση των συμβόλων, το επόμενο βήμα είναι να συγκρίνουμε τα αναγνωρισμένα σύμβολα με τα αρχικά σύμβολα που στάλθηκαν. Με αυτόν τον τρόπο, μπορούμε να υπολογίσουμε πόσα λάθη (σφάλματα) έγιναν κατά τη διάρκεια της μετάδοσης του σήματος, κυρίως λόγω θορύβου και άλλων παρεμβολών.Τα αναγνωρισμένα σύμβολα αποθηκεύονται στο διάνυσμα detected_symbols. Τα αρχικά σύμβολα (όπως δημιουργήθηκαν στην αρχή του κώδικα, με την εντολή symbols = A(randi([1, M], 1, N));) αποθηκεύονται στο διάνυσμα symbols.Ο αριθμός των σφαλμάτων υπολογίζεται συγκρίνοντας τα δύο διανύσματα, δηλαδή συγκρίνοντας τα αρχικά σύμβολα με τα αναγνωρισμένα σύμβολα:

$$errors = \sum(detected\_symbols \neq symbols)$$

Τελικά, η συνάρτηση sum μετρά τον αριθμό των διαφορών (σφαλμάτων) μεταξύ των αντίστοιχων στοιχείων των διανυσμάτων detected_symbols και symbols. Αν τα δύο στοιχεία είναι διαφορετικά, το αποτέλεσμα είναι 1 (δηλαδή, σφάλμα). Αν τα δύο στοιχεία είναι τα ίδια, το αποτέλεσμα είναι 0 (δεν υπάρχει σφάλμα).

```
% Φωρατής
detected_symbols = zeros(1, N);
for i = 1:N
    [~, idx] = min(abs(A - r(i)));  % Χρησιμοποιούμε το A αντί του Am
    detected_symbols(i) = A(idx);  % Ανίχνευση του συμβόλου
end


% Υπολογισμός σφαλμάτων
errors = sum(detected_symbols ~= symbols);
```

Αυτό είναι το αποτέλεσμά μας:




```
>> MPAM
Total Errors: 486
```

ΠΑΡΑΤΗΡΗΣΗ: Τα "original symbols" είναι τα διακριτά επίπεδα πλάτους που έχουν παραχθεί από τη διαδικασία διαμόρφωσης και αντιπροσωπεύουν την πληροφορία που θέλουμε να μεταδοθεί. Σε M-PAM, τα σύμβολα ανήκουν σε ένα σύνολο από M διακριτές τιμές, οι οποίες είναι συμμετρικά τοποθετημένες γύρω από το μηδέν. Εδώ τα σύμβολα φαίνονται σε επίπεδα όπως −4,−2,0,+2,+4 για M=8, όπου κάθε επίπεδο κωδικοποιεί διαφορετική πληροφορία. Το πρώτο γράφημα απεικονίζει την αρχική ακολουθία συμβόλων πριν από τη μετάδοση μέσω του καναλιού. Το "received signal" αντιπροσωπεύει το σήμα όπως λαμβάνεται από τον δέκτη μετά τη μετάδοση μέσω του καναλιού. Το κανάλι εισάγει θόρυβο (συνήθως λευκό Gaussian θόρυβο), το οποίο προκαλεί απόκλιση από τις αρχικές τιμές των συμβόλων. Μπορεί να παρατηρηθεί ότι τα σύμβολα έχουν μετατοπιστεί ελαφρώς λόγω της παρουσίας του θορύβου, ενώ διατηρούν τη γενική δομή τους. Ο θόρυβος μπορεί να προκαλέσει σύγχυση, ιδιαίτερα όταν οι τιμές των συμβόλων είναι κοντά μεταξύ τους. Τα "detected symbols" είναι το αποτέλεσμα της διαδικασίας αποδιαμόρφωσης και ανίχνευσης στον δέκτη. Ο δέκτης εκτελεί μια πράξη ποσοτικοποίησης (quantization), αντιστοιχίζοντας το λαμβανόμενο σήμα στο πιο κοντινό διακριτό επίπεδο συμβόλων. Εάν ο θόρυβος έχει μετακινήσει ένα σύμβολο πέρα από το όριο μεταξύ δύο επιπέδων, τότε αυτό μπορεί να οδηγήσει σε σφάλμα ανίχνευσης. Τα πράσινα σύμβολα είναι οι τελικές τιμές που ο δέκτης αναγνωρίζει ως τις πιο πιθανές.
Βασικά Σημεία της Υλοποίησης:
Τα σύμβολα διαμορφώνονται σε MM διακριτά επίπεδα πλάτους. Το M-PAM είναι απλή διαμόρφωση και χρησιμοποιεί τα επίπεδα ακεραίων για την αναπαράσταση της πληροφορίας.
Κατά τη μετάδοση μέσω του καναλιού, προστίθεται θόρυβος Gaussian (AWGN - Additive White Gaussian Noise), που προκαλεί διαταραχή στα αρχικά σύμβολα.
Ο δέκτης εφαρμόζει ένα όριο απόφασης (decision boundary) για να ανιχνεύσει το πιο κοντινό σύμβολο. Τα σφάλματα συμβαίνουν όταν ο θόρυβος μετακινεί το λαμβανόμενο σήμα σε διαφορετική περιοχή απόφασης.
Σφάλματα εμφανίζονται όταν τα detected symbols (πράσινο) διαφέρουν από τα original symbols (μπλε). Τα σφάλματα αυτά μετρώνται είτε σε επίπεδο bit (BER) είτε σε επίπεδο συμβόλων (SER).

(ii). Αρχικοποιούμε μια μεταβλητή τύπου struct για να αποθηκευτούν τα αποτελέσματα για κάθε τιμή του M. Μετά για κάθε τιμή του M (πλήθος επιπέδων διαμόρφωσης M-PAM), υπολογίζουμε τον αριθμό των bits ανά σύμβολο k= $\log_2 M$. Η προσαρμογή του πλήθους των bits είναι απαραίτητη για να διασφαλιστεί ότι ο αριθμός των bits που θα χρησιμοποιηθούν για τη διαμόρφωση μπορεί να χωριστεί σε ομάδες k bits. Εάν ο αριθμός των αρχικών bits δεν είναι πολλαπλάσιο του k, κάποια bits θα περισσεύουν και δεν θα μπορούν να αντιστοιχιστούν σε σύμβολο. Συγκεκριμένα, η διαίρεση Lb/k δίνει τον αριθμό των πλήρων ομάδων k bits που χωράνε στα Lb (συνολικό πλήθος των αρχικών bits). Η συνάρτηση floor διατηρεί μόνο το ακέραιο μέρος της διαίρεσης. Αυτό σημαίνει ότι αν υπάρχουν περισσευόμενα bits που δεν σχηματίζουν πλήρη ομάδα k, αυτά αγνοούνται. Το αποτέλεσμα floor(Lb/k)·k δίνει το μεγαλύτερο πλήθος bits που είναι πολλαπλάσιο του k και μπορεί να ομαδοποιηθεί χωρίς να περισσέψει τίποτα. Στη συνέχεια, δημιουργείται μια τυχαία ακολουθία από Lb_adjusted bits, η οποία πλέον είναι πολλαπλάσιο του k. Αυτή η ακολουθία μπορεί να ομαδοποιηθεί και να αντιστοιχιστεί σε σύμβολα χωρίς προβλήματα. Με αυτόν τον τρόπο δεν υπάρχει η ανάγκη αντιμετώπισης περισσευούμενων bits.

```
% Αρχικοποίηση αποτελεσμάτων
results = struct();

for M = M_values
    k = log2(M); % Bits per symbol

    % Προσαρμογή του πλήθους των bits ώστε να είναι πολλαπλάσιο του k
    Lb_adjusted = floor(Lb / k) * k; % Πολλαπλάσιο του k
    bits = randi([0, 1], Lb_adjusted, 1); % Τυχαία ακολουθία bits
```

Προχωράμε στη διαδικασία ομαδοποίησης των bits σε σύμβολα, ξεκινώντας με τον υπολογισμό του αριθμού των συμβόλων. Το πλήθος των bits Lb_adjusted που έχουν ήδη προσαρμοστεί (ώστε να είναι πολλαπλάσιο του k) διαιρείται με k, το πλήθος των bits ανά σύμβολο και το αποτέλεσμα num_symbols είναι ο αριθμός των συμβόλων που θα δημιουργηθούν. Η εντολή reshape αναδιατάσσει τα Lb_adjusted bits σε έναν πίνακα με k γραμμές και num_symbols στήλες. Κάθε στήλη περιέχει k bits, τα οποία αντιστοιχούν σε ένα σύμβολο. Μετατρέπουμε τις δυαδικές ομάδες σε ακέραια σύμβολα δημιουργώντας για αρχή έναν πίνακα symbols για την αποθήκευση των ακέραιων τιμών των συμβόλων. Για όλο τον αριθμό συμβόλων και για κάθε στήλη της bit_groups, που αντιπροσωπεύει ένα δυαδικό αριθμό , γίνεται μετατροπή από δυαδικό σε δεκαδικό. Η έκφραση 2 ^(k−1:−1:0) δημιουργεί έναν πίνακα με τις δυνάμεις του 2, ανάλογα με τη θέση των bits. Ο πίνακας bit_groups(:,i)′ είναι οριζόντιος για σωστό πολλαπλασιασμό και το γινόμενο bit_groups(:,i)′.∗2.(k−1:−1:0) υπολογίζει τη δεκαδική τιμή του αριθμού. Τέλος, η συνάρτηση sum αθροίζει τις δεκαδικές τιμές για να υπολογίσει το τελικό σύμβολο. Το αποτέλεσμα της διαδικασίας είναι ένας πίνακας symbols, όπου κάθε στοιχείο αντιπροσωπεύει ένα σύμβολο. Συνεχίζουμε διαμορφώνοντας το M-PAM με 3 απλά βήματα. Ορίζουμε την ενέργεια ανά σύμβολο Es ως 1. Υπολογίζουμε το πλάτος A για M-PAM με τρόπο ώστε η ενέργεια των συμβόλων να παραμένει σταθερή ακολουθώντας τον μαθηματικό τύπο:

$$A=\sqrt{\frac{3*Es}{M\hat{}2}}.$$

Τα σύμβολα διαμορφώνονται σε M-PAM σύμφωνα με τη μαθηματική σχέση x=A(2m−M+1). Τέλος, δημιουργούμε πίνακες για το BER (Bit Error Rate) και το SER (Symbol Error Rate) για κάθε τιμή SNR_dB, που αρχικοποιούνται με μηδενικά.

```
% Βήμα 2: Αντιστοίχιση bits σε σύμβολα
    num_symbols = Lb_adjusted / k; % Αριθμός συμβόλων
    bit_groups = reshape(bits, k, []); % Ομαδοποίηση σε ομάδες k bits
    symbols = zeros(1, num_symbols);
    for i = 1:num_symbols
        symbols(i) = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0)); % Υπολογισμός συμβόλου
    end

    % Βήμα 3: Διαμόρφωση M-PAM
    Es = 1; % Ενέργεια ανά σύμβολο
    A = sqrt(3 * Es / (M^2 - 1)); % Πλάτος
    pam_symbols = A * (2 * symbols - M + 1); % Διαμόρφωση PAM

    % Αρχικοποίηση BER και SER
    BER = zeros(size(SNR_dB));
    SER = zeros(size(SNR_dB));
```

Θα προσθέσουμε θόρυβο AWGN (Additive White Gaussian Noise) στο διαμορφωμένο σήμα. Υποόγίζουμε την διασπορά θορύβου από την σχέση:

$$\sigma^2 = \frac{Es}{10^{\left(\frac{SNR_{dB}}{10}\right)}}$$

Όπου Es είναι η ενέργεια ανά σύμβολο, και η τιμή SNR_dB είναι η τρέχουσα SNR σε dB. Η SNR μετρά την ισχύ του σήματος σε σχέση με τον θόρυβο. Η SNR σε dB είναι ο λόγος της ισχύος του σήματος προς την ισχύ του θορύβου:

$$SNR_{dB} = 10\log_{10}\left(\frac{Psignal}{Pnoise}\right)$$

Το SNR_dB(idx) είναι η αναλογία σήματος προς θόρυβο (Signal-to-Noise Ratio) σε μονάδες dB. Η συνάρτηση randn παράγει τυχαίους αριθμούς από μια κανονική κατανομή με μέση τιμή 0 και διασπορά 1. Ο θόρυβος δημιουργείται με τη μορφή noise=variance×randn. Εδώ, η διασπορά προσαρμόζεται ώστε να αντιστοιχεί στη σωστή ισχύ θορύβου για την εκάστοτε SNR. Το παρατηρούμενο σήμα που φτάνει στον δέκτη είναι το διαμορφωμένο σήμα με προσθήκη του θορύβου received_signal=pam_symbols+noise. Αυτό το σήμα είναι αυτό που θα επεξεργαστεί ο δέκτης, και περιέχει την παραμόρφωση λόγω του θορύβου. Επίσης πρέπει να αναγνωρίσουμε ταδιαμορφωμένα σύμβολα από το παρατηρούμενο σήμα που περιέχει θόρυβο. Ο στόχος είναι να προσδιορίσουμε ποιο σύμβολο (από τα M διαθέσιμα σύμβολα) είναι το πιο πιθανό να έχει αποσταλεί, δεδομένου του θορύβου που προστέθηκε. Αποφασίζουμε μέσω κανονικοποίησης, δηλαδή το σήμα με θόρυβο received_signal διαμοιράζεται με το πλάτος A για να φέρει το σήμα στον σωστό εύρος τιμών. Στη συνέχεια, προστίθεται M−1(το μέγιστο σύμβολο), και διαιρείται με 2 για να μετατραπεί σε ακέραιες τιμές. Η συνάρτηση round χρησιμοποιείται για να στρογγυλοποιήσει το αποτέλεσμα στην κοντινότερη ακέραια τιμή. Αυτό ουσιαστικά σημαίνει ότι προσδιορίζεται το πιο κοντινό σύμβολο (στο επίπεδο απόφασης). Και η εντολή max(min(detected_symbols, M-1), 0) περιορίζει τις τιμές του detected_symbols ώστε να βρίσκονται εντός του εύρους [0, M-1], το οποίο αντιστοιχεί στα έγκυρα επίπεδα σύμβολων. Η διαδικασία της αντίστροφης αντιστοίχισης συμβόλων σε bits είναι απαραίτητη για την αποκωδικοποίηση των σημάτων, ώστε να ανακτηθεί η αρχική ακολουθία των bits που αποστάλθηκε πριν από τη διαμόρφωση. Ουσιαστικά ομαδοποιούμε τα bits σε σύμβολα. Δημιουργείται ένας αρχικά μηδενικός πίνακας detected_bits μεγέθους Lb_adjusted για την αποθήκευση των ανιχνευμένων bits. Η τιμή Lb_adjusted είναι το πλήθος των bits που έχουν προσαρμοστεί ώστε να είναι πολλαπλάσιο του k. Αυτός ο πίνακας θα περιέχει τη δυαδική αναπαράσταση των ανιχνευμένων συμβόλων. Για κάθε ανιχνευμένο σύμβολο η τιμή του ανιχνευμένου συμβόλου detected_symbols(j) ανακτάται και αποθηκεύεται στη μεταβλητή dec_value. Κάθε σύμβολο έχει k bits, οπότε για να ανακτήσουμε τα k bits που αντιστοιχούν στο σύμβολο, χρησιμοποιούμε την mod(floor(dec_value / 2^(k-b)), 2) για να υπολογίσουμε το b-οστό bit από τα k bits του συμβόλου. Η τιμή του dec_value διαιρείται με την κατάλληλη δύναμη του 2, και η συνάρτηση floor αφαιρεί την κλασματική τιμή, κρατώντας μόνο την ακέραια τιμή.
Εφαρμόζεται το υπόλοιπο της διαίρεσης με το 2 για να εξάγουμε το bit (0 ή 1). Αυτός ο υπολογισμός γίνεται για κάθε bit του συμβόλου, και τα αποτελέσματα αποθηκεύονται στον πίνακα detected_bits.

```
for idx = 1:length(SNR_dB)
        % Βήμα 4: Προσθήκη θορύβου
        noise_variance = Es / (10^(SNR_dB(idx)/10)); % Διασπορά θορύβου
        noise = sqrt(noise_variance) * randn(size(pam_symbols)); % Θόρυβος AWGN
        received_signal = pam_symbols + noise; % Σήμα με θόρυβο

        % Βήμα 5: Απόφαση συμβόλων
        detected_symbols = round((received_signal / A + M - 1) / 2); % Απόφαση
        detected_symbols = max(min(detected_symbols, M-1), 0); % Περιορισμός εύρους

        % Αντίστροφη αντιστοίχιση συμβόλων σε bits
        detected_bits = zeros(Lb_adjusted, 1);
        for j = 1:num_symbols
            dec_value = detected_symbols(j);
            for b = 1:k
                detected_bits((j-1)*k + b) = mod(floor(dec_value / 2^(k-b)), 2);
            end
        end
```

Υπολογίζουμε το BER (Bit Error Rate) και το SER (Symbol Error Rate). Το BER είναι η αναλογία των σφαλμάτων στα bits, δηλαδή το ποσοστό των bits που αναγνωρίστηκαν λανθασμένα σε σχέση με τον συνολικό αριθμό των bits που αποστάλθηκαν. Συγκρίνουμε την πρωτότυπη ακολουθία bits (bits) με την ανιχνευμένη ακολουθία bits (detected_bits), και επιστρέφεται ένας πίνακας λογικών τιμών (1 για σφάλμα, 0 για σωστή αντιστοίχιση). Ο υπολογισμός sum(bits ~= detected_bits) μετράει τον αριθμό των σφαλμάτων, δηλαδή το πόσα bits είναι διαφορετικά μεταξύ των αποσταλθέντων και των ανιχνευμένων. Ο αριθμός των σφαλμάτων διαιρείται με το συνολικό πλήθος των bits που αποστάλθηκαν (Lb_adjusted), το οποίο είναι το συνολικό μέγεθος της ακολουθίας των bits (προσαρμοσμένο ώστε να είναι πολλαπλάσιο του k). Το αποτέλεσμα είναι το BER για τη δεδομένη τιμή της SNR (αναλογία σήματος προς θόρυβο) σε dB.
Το SER είναι η αναλογία των σφαλμάτων στα σύμβολα, δηλαδή το ποσοστό των συμβόλων που αναγνωρίστηκαν λανθασμένα σε σχέση με τον συνολικό αριθμό των συμβόλων που αποστάλθηκαν. Με το ίδιο σκεπτικό όπως για το BER συγκρίνουμε τα αρχικά σύμβολα (symbols) με τα ανιχνευμένα σύμβολα (detected_symbols), και επιστρέφεται έναν πίνακα λογικών τιμών (1 για σφάλμα, 0 για σωστή αντιστοίχιση). Ο υπολογισμός sum(symbols ~= detected_symbols) μετράει τον αριθμό των σφαλμάτων στα σύμβολα, δηλαδή το πόσα σύμβολα αναγνωρίστηκαν λανθασμένα. Τέλος, ο αριθμός των σφαλμάτων διαιρείται με το συνολικό πλήθος των συμβόλων που αποστάλθηκαν (num_symbols), το οποίο είναι το μέγεθος της ακολουθίας των συμβόλων (και ισούται με το συνολικό πλήθος των bits διαιρεμένο με το k, επειδή κάθε σύμβολο αντιστοιχεί σε k bits). Το αποτέλεσμα είναι επίσης το SER για τη δεδομένη τιμή της SNR (αναλογία σήματος προς θόρυβο) σε dB. Για αποθήκευση των διαφορετικών αποτελεσμάτων για κάθε του M, που είναι το μέγεθος του σύμβολου στο σύστημα M-PAM, χρησιμοποιείται μια δομή results.

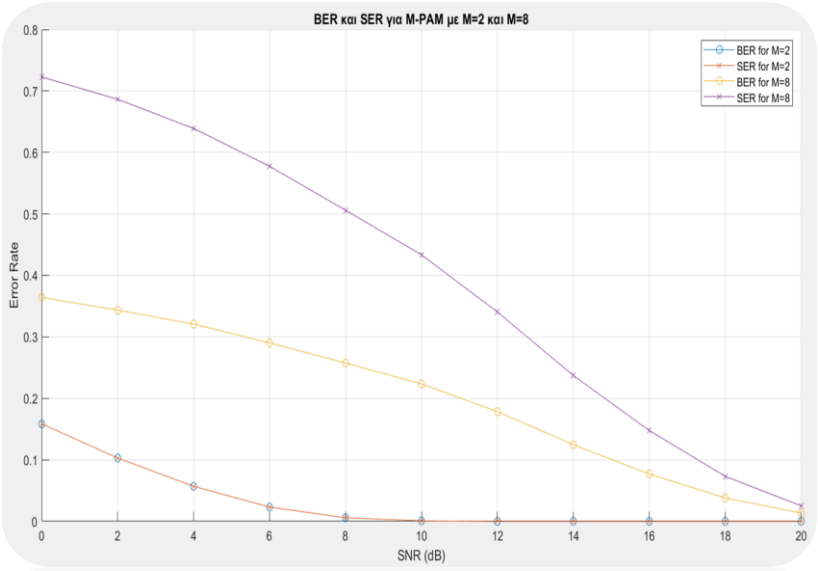
results(M).SNR_dB = SNR_dB; :Για κάθε τιμή του M, αποθηκεύεται η τιμή του SNR σε dB σε ένα πεδίο της δομής results. Η μεταβλητή SNR_dB περιέχει τις διάφορες τιμές της αναλογίας σήματος προς θόρυβο για τις οποίες υπολογίζονται το BER και το SER. Το πεδίο SNR_dB θα περιέχει την αντίστοιχη τιμή του SNR σε dB που χρησιμοποιήθηκε για να υπολογιστούν τα αποτελέσματα για την τρέχουσα τιμή του M.

results(M).BER = BER; :Στο πεδίο BER της δομής results(M), αποθηκεύεται το Bit Error Rate (BER) για τη δεδομένη τιμή του MM και για τις διαφορετικές τιμές του SNR. Η μεταβλητή BER περιέχει έναν πίνακα με τις τιμές του BER για κάθε τιμή του SNR (για τις διάφορες τιμές θορύβου και σήματος).

results(M).SER = SER; :Στο πεδίο SER της δομής results(M), αποθηκεύεται το Symbol Error Rate (SER) για την ίδια τιμή του MM και για τις διαφορετικές τιμές του SNR. Η μεταβλητή SER περιέχει έναν πίνακα με τις τιμές του SER για κάθε τιμή του SNR, που δείχνει πόσο συχνά τα σύμβολα αναγνωρίζονται λανθασμένα.
Άρα, προκύπτουν τρεις πίνακες, ένας με τις διάφορες τιμές του SNR σε dB, ένας με τις τιμές του Bit Error Rate (BER) για κάθε τιμή του SNR και ένας με τις τιμές του Symbol Error Rate (SER) για κάθε τιμή του SNR.

```
  % Υπολογισμός BER και SER
        BER(idx) = sum(bits ~= detected_bits) / Lb_adjusted;
        SER(idx) = sum(symbols ~= detected_symbols) / num_symbols;
    end

    % Αποθήκευση αποτελεσμάτων
    results(M).SNR_dB = SNR_dB;
    results(M).BER = BER;
    results(M).SER = SER;
end
```

Όταν τρέξουμε τον παραπάνω κώδικα έχουμε τα εξής αποτελέσματα:



Αποτελέσματα για M=2:

SNR_dB	BER_M2
0	0.15969
2	0.10349
4	0.05645
6	0.02301
8	0.00598
10	0.00075
12	2e-05
14	0
16	0
18	0
20	0

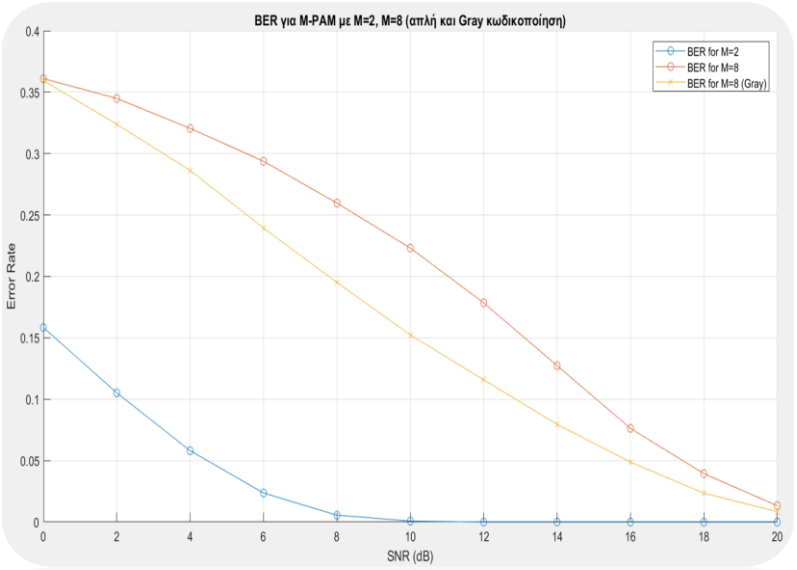
Αποτελέσματα για M=8 (Απλή Κωδικοποίηση):

SNR_dB	BER_M8_Simple
0	0.36351
2	0.34345
4	0.31993
6	0.29299
8	0.25999
10	0.22438
12	0.17737
14	0.12706
16	0.075701
18	0.03773
20	0.0132

Τώρα πάμε στην κωδικοποίηση Gray για το σύστημα M-PAM για M=8. Η κωδικοποίηση Gray είναι μια μέθοδος αντιστοίχισης bits σε σύμβολα, στην οποία μόνο ένα bit αλλάζει για κάθε διαδοχικό σύμβολο. Η διαφορά εντοπίζεται στον τρόπο που επιτυγχάνεται η αντιστοίχιση των bits στα σύμβολα. Δημιουργούμε τον πίνακα symbols_gray που περιέχει το πλήθος των συμβόλων και τον αρχικοποιούμε πάλι με μηδενικά. Ο πίνακας gray_map χρησιμοποιείται για να γίνει η αντιστοίχιση Gray. Η συνάρτηση bitxor εφαρμόζεται για να παράγει την κωδικοποιημένη τιμή Gray για κάθε συμβολοποιημένο αριθμό. Στην συνέχεια, γίνεται αντιστοίχιση του δυαδικού αριθμού στους Gray κωδικοποιημένους αριθμούς χρησιμοποιώντας την εντολή find(gray_map == binary_value). Η διαμόρφωση γίνεται με τα σύμβολα που έχουν κωδικοποιηθεί μέσω Gray και ακολουθεί την ίδια μορφή με την κανονική, αλλά τα σύμβολα που χρησιμοποιούνται είναι τα κωδικοποιημένα Gray σύμβολα (symbols_gray) αντί για τα κανονικά σύμβολα (symbols). Ακόμη στην αντίστροφη αντιστοίχιση Gray όταν ανιχνεύονται τα σύμβολα, γίνεται αρχικά η αντιστοίχιση των ανιχνευμένων συμβόλων σε Gray κωδικοποιημένες τιμές με την εντολή gray_value = gray_map(detected_symbols(j) + 1). Στη συνέχεια, γίνεται η αντίστροφη διαδικασία μετατροπής των Gray κωδικοποιημένων τιμών σε δυαδικά bits μέσω της εξίσωσης mod(floor(gray_value / 2^(k-b)), 2). Τέλος, για τον υπολογισμό του BER χρησιμοποιούμε τα Gray-κωδικοποιημένα σύμβολα και τα ανιχνευμένα bits μετά την αντίστροφη αντιστοίχιση Gray. Ο τρόπος υπολογισμού του BER παραμένει ο ίδιος.

```
M = 8;
k = log2(M);
Lb_adjusted = floor(Lb / k) * k;
bits = randi([0, 1], Lb_adjusted, 1);
num_symbols = Lb_adjusted / k;
bit_groups = reshape(bits, k, []);
symbols_gray = zeros(1, num_symbols);

% Αντιστοίχιση Gray
gray_map = bitxor((0:M-1)', floor((0:M-1)'/2));
for i = 1:num_symbols
    binary_value = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0));
    symbols_gray(i) = find(gray_map == binary_value) - 1;
end

% Διαμόρφωση με Gray
pam_symbols_gray = A * (2 * symbols_gray - M + 1);
BER_gray = zeros(size(SNR_dB));

for idx = 1:length(SNR_dB)
    noise_variance = Es / (10^(SNR_dB(idx)/10));
    noise = sqrt(noise_variance) * randn(size(pam_symbols_gray));
    received_signal = pam_symbols_gray + noise;

    % Απόφαση Gray
    detected_symbols = round((received_signal / A + M - 1) / 2);
    detected_symbols = max(min(detected_symbols, M-1), 0);

    % Αντιστοίχιση Gray
    detected_bits = zeros(Lb_adjusted, 1);
    for j = 1:num_symbols
        gray_value = gray_map(detected_symbols(j) + 1);
        for b = 1:k
            detected_bits((j-1)*k + b) = mod(floor(gray_value / 2^(k-b)), 2);
        end
    end

    % Υπολογισμός BER Gray
    BER_gray(idx) = sum(bits ~= detected_bits) / Lb_adjusted;
end
```

Τώρα έχουμε το εξής αποτέλεσμα:



Αποτελέσματα για M=8 (Κωδικοποίηση Gray):

SNR_dB	BER_M8_Gray
0	0.36029
2	0.32691
4	0.28507
6	0.23987
8	0.19377
10	0.15265
12	0.1137
14	0.081381
16	0.050171
18	0.02347
20	0.0087101

ΠΑΡΑΤΗΡΗΣΗ:
Οριζόντιος άξονας (SNR σε dB): Αντιπροσωπεύει την αναλογία σήματος προς θόρυβο σε δεκαδικά λογαριθμική κλίμακα. Η αύξηση του SNR σημαίνει λιγότερος θόρυβος στο σήμα, κάτι που γενικά οδηγεί σε χαμηλότερο BER.

Κατακόρυφος άξονας (Error Rate): Βρίσκεται σε λογαριθμική κλίμακα για το BER και η μικρότερη τιμή BER σημαίνει καλύτερη απόδοση.

Το M=2 είναι η περίπτωση 2-PAM, που ισοδυναμεί με την απλή δυαδική διαμόρφωση (Binary PAM). Το BER για M=2 είναι χαμηλότερο σε όλες τις τιμές του SNR, κάτι που αναμενόταν, καθώς έχει τη μεγαλύτερη απόσταση μεταξύ των επιπέδων σήματος και είναι λιγότερο ευαίσθητο στον θόρυβο.
Το M=8 αντιπροσωπεύει την 8-PAM διαμόρφωση. Παρατηρείται υψηλότερο BER σε χαμηλά επίπεδα SNR σε σχέση με το M=2, καθώς τα επίπεδα σήματος είναι πιο κοντά μεταξύ τους και είναι ευκολότερο για τον θόρυβο να προκαλέσει λάθος. Όταν το SNR αυξάνεται (π.χ., πάνω από 16 dB), το BER μειώνεται σημαντικά.

Για την Gray κωδικοποίηση με M=8 ελαχιστοποιούνται τα λάθη bit αντιστοιχίζοντας διαδοχικά επίπεδα σήματος έτσι ώστε να διαφέρουν μόνο κατά ένα bit. Παρατηρείται χαμηλότερο BER σε σχέση με την απλή 8-PAM διαμόρφωση (κόκκινη καμπύλη), ειδικά σε μεσαία και χαμηλά επίπεδα SNR, δείχνοντας την απόδοση της Gray κωδικοποίησης. Η Gray κωδικοποίηση βελτιώνει την απόδοση του συστήματος για M=8, ειδικά σε χαμηλά και μεσαία επίπεδα SNR. Αυτό δείχνει τη χρησιμότητά της σε συστήματα υψηλότερης πολυπλοκότητας διαμόρφωσης.

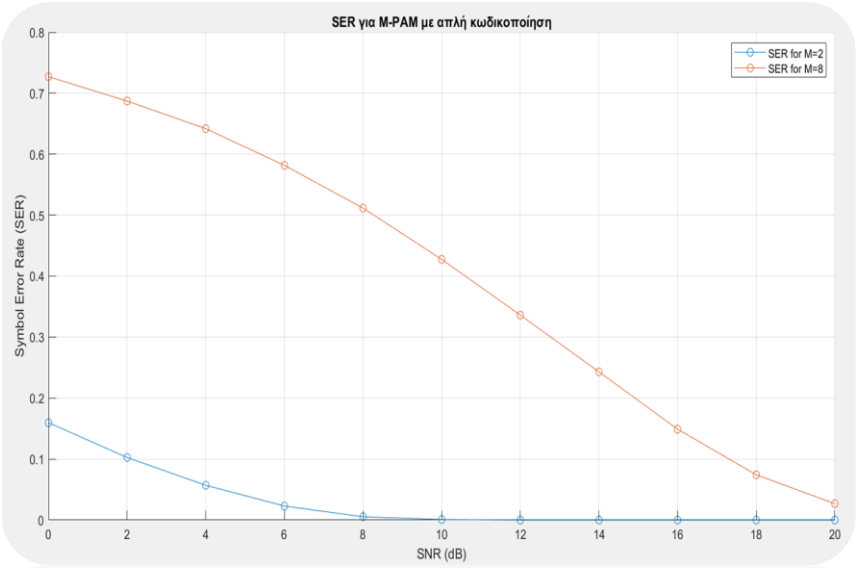
(iii). Τώρα πρέπει να μετρήσουμε την πιθανότητα σφάλματος συμβόλου αντί για bit. Η πιθανότητα σφάλματος συμβόλου εκφράζει το ποσοστό των λανθασμένων αποφάσεων μεταξύ των πραγματικών και των ανιχνευμένων συμβόλων.Για αρχή δημιουργούμε μια δομή results, που θα αποθηκεύει τα αποτελέσματα του υπολογισμού SER για κάθε τιμή του M. Ξεκινάμε μία επανάληψη για κάθε τιμή του M από τον πίνακα M_values, δηλαδή θα υπολογίσουμε το SER για διάφορες διαμορφώσεις του M-PAM. Υπολογίζουμε τον αριθμό των bits ανά σύμβολο k, ο οποίος είναι ο λογάριθμος του M στη βάση 2. Αυτό σημαίνει ότι για κάθε σύμβολο της διαμόρφωσης M-PAM, χρειάζονται k bits. Ορίζουμε πάλι το Lb_adjusted και δημιουργούμε ξανά με την συνάρτηση randi μια τυχαία ακολουθία bits μεγέθους Lb_adjusted (τα bits θα είναι είτε 0 είτε 1). Βρίσκουμε τον αριθμό των συμβόλων num_symbols που πρέπει να δημιουργηθούν και στη συνέχεια, η ακολουθία των bits αναδιοργανώνεται σε ομάδες των k bits χρησιμοποιώντας τη συνάρτηση reshape, και αρχικοποιείται ο πίνακας symbols για την αποθήκευση των συμβόλων. Για κάθε ομάδα k bits, υπολογίζεται το αντίστοιχο σύμβολο χρησιμοποιώντας τη δυαδική

αντιστοίχιση. Κάθε ομάδα bits μετατρέπεται σε αριθμό μέσω του τύπου sum(bit_groups(:, i)' .* 2.^(k-1:-1:0)). Ορίζουμε ενέργεια και πλάτος για την διαμόρφωση του PAM και ο πίνακας SER αρχικοποιείται για να αποθηκεύσει τις τιμές του Symbol Error Rate για κάθε τιμή SNR_dB. Έπειτα ξεκινά η επανάληψη για κάθε τιμή του λόγου SNR σε dB. Εκεί υπολογίζεται η διασπορά του θορύβου βάσει του SNR με τη φόρμουλα noise_variance = Es / (10^(SNR_dB(idx)/10)), δημιουργείται ο θόρυβος noise χρησιμοποιώντας τη συνάρτηση randn και προστίθεται στο διαμορφωμένο σήμα για να δημιουργηθεί το received_signal με θόρυβο. Μετά το received_signal διαιρείται με το A και προσαρμόζεται για να γίνει απόφαση για τα ανιχνευθέντα σύμβολα detected_symbols και εφαρμόζεται περιορισμός της τιμής των ανιχνευμένων συμβόλων γίνεται έτσι ώστε να βρίσκονται στο εύρος [0, M-1]. Τα αποτελέσματα για την τρέχουσα τιμή του M αποθηκεύονται στην δομή

results.

```
% Αρχικοποίηση αποτελεσμάτων
results = struct();

for M = M_values
    k = log2(M); % Bits per symbol

    % Προσαρμογή του πλήθους των bits ώστε να είναι πολλαπλάσιο του k
    Lb_adjusted = floor(Lb / k) * k; % Πολλαπλάσιο του k
    bits = randi([0, 1], Lb_adjusted, 1); % Τυχαία ακολουθία bits

    % Βήμα 2: Αντιστοίχιση bits σε σύμβολα
    num_symbols = Lb_adjusted / k; % Αριθμός συμβόλων
    bit_groups = reshape(bits, k, []); % Ομαδοποίηση σε ομάδες k bits
    symbols = zeros(1, num_symbols);
    for i = 1:num_symbols
        symbols(i) = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0)); % Υπολογισμός συμβόλου
    end

    % Βήμα 3: Διαμόρφωση M-PAM
    Es = 1; % Ενέργεια ανά σύμβολο
    A = sqrt(3 * Es / (M^2 - 1)); % Πλάτος
    pam_symbols = A * (2 * symbols - M + 1); % Διαμόρφωση PAM

    % Αρχικοποίηση SER
    SER = zeros(size(SNR_dB));

    for idx = 1:length(SNR_dB)
        % Βήμα 4: Προσθήκη θορύβου
        noise_variance = Es / (10^(SNR_dB(idx)/10)); % Διασπορά θορύβου
        noise = sqrt(noise_variance) * randn(size(pam_symbols)); % Θόρυβος AWGN
        received_signal = pam_symbols + noise; % Σήμα με θόρυβο

        % Βήμα 5: Απόφαση συμβόλων
        detected_symbols = round((received_signal / A + M - 1) / 2); % Απόφαση
        detected_symbols = max(min(detected_symbols, M-1), 0); % Περιορισμός εύρους

        % Υπολογισμός SER
        SER(idx) = sum(symbols ~= detected_symbols) / num_symbols;
    end

    % Αποθήκευση αποτελεσμάτων
    results(M).SNR_dB = SNR_dB;
    results(M).SER = SER;
```

Το αποτέλεσμα είναι το εξής:




```
>> SER_M_PAM_Simulation
Αποτελέσματα για M = 2:
    SNR_dB      SER
    ______    _______

       0      0.15966
       2      0.10253
       4      0.05696
       6      0.02296
       8      0.00538
      10       0.0009
      12        3e-05
      14            0
      16            0
      18            0
      20            0

Αποτελέσματα για M = 8:
    SNR_dB      SER
    ______    _______

       0      0.72709
       2       0.6871
       4       0.6418
       6      0.58162
       8      0.51154
      10      0.42747
      12      0.33603
      14      0.24303
      16      0.14895
      18     0.074191
      20      0.02685
```

ΠΑΡΑΤΗΡΗΣΗ:

Το M=2, όπως και στο προηγούμενο BER διάγραμμα, έχει το χαμηλότερο SER, καθώς έχει τη μεγαλύτερη απόσταση μεταξύ των επιπέδων σήματος, άρα είναι λιγότερο ευαίσθητο στον θόρυβο. Στην περιοχή χαμηλού SNR (0−6 dB), το SER μειώνεται από περίπου 0.2 σε σχεδόν 0.01, ακολουθώντας εκθετική πτώση με την αύξηση του SNR. Σε υψηλές τιμές SNR (>10 dB), το SER γίνεται εξαιρετικά μικρό (πρακτικά μηδενικό), κάτι που δείχνει ότι η δυαδική διαμόρφωση M=2 είναι ανθεκτική στο θόρυβο. Υπάρχουν μόνο δύο επίπεδα σήματος, και συνεπώς τα όρια ανίχνευσης είναι σαφή και το σύστημα είναι λιγότερο επιρρεπές σε λάθη λόγω θορύβου.

Το M=8 έχει υψηλότερο SER από το M=2 σε όλες τις τιμές του SNR, κάτι που αναμενόταν λόγω της μικρότερης απόστασης μεταξύ των συμβόλων στην 8-PAM διαμόρφωση. Στην περιοχή SNR=0−6 dB, το SER μειώνεται πιο αργά σε σχέση με το M=2. Για παράδειγμα, για SNR=4 dB, το SER είναι περίπου 0.4, ενώ για SNR=10 dB, το SER είναι περίπου 0.1. Σε υψηλότερα SNR (>16 dB), το SER μειώνεται σε χαμηλά επίπεδα, αλλά δεν πλησιάζει τόσο γρήγορα το μηδέν όσο στην περίπτωση M=2. Παρατηρείται σταδιακή μείωση του SER καθώς το SNR αυξάνεται. Γενικά, στην 8-PAM, υπάρχουν περισσότερα επίπεδα σήματος, γεγονός που σημαίνει ότι τα όρια ανίχνευσης είναι πιο κοντά μεταξύ τους. Αυτό καθιστά το σύστημα πιο ευαίσθητο στον θόρυβο, ιδιαίτερα σε χαμηλά SNR.

Το SER (Symbol Error Rate) είναι συνήθως μεγαλύτερο από το BER (Bit Error Rate), και αυτό οφείλεται στο ότι μετράει σφάλματα σε επίπεδο συμβόλων, ενώ το BER λαμβάνει υπόψη τα bits που περιέχονται σε κάθε σύμβολο. Συγκεκριμένα, ένα σφάλμα σε επίπεδο συμβόλου (SER) μπορεί να οδηγήσει σε σφάλματα σε πολλαπλά bits. Αυτό αυξάνει την πιθανότητα σφάλματος σε bits, αλλά η τιμή του BER είναι χαμηλότερη από το SER επειδή μετράει την πιθανότητα σφάλματος ανά bit, όχι ανά σύμβολο.

Όλοι οι κώδικες που χρησιμοποιήθηκαν σε αυτή την εργασία βρίσκονται στο παρακάτω Παράρτημα.

ΜΕΡΟΣ Α

PCM_quantizer.m

```
function [xq, centers] = my_quantizer(x, N, min_value, max_value)
    % Περιορισμός του σήματος στις αποδεκτές τιμές [min_value, max_value]
    x = max(min(x, max_value), min_value);

    % Υπολογισμός του αριθμού επιπέδων κβαντισμού
    L = 2^N;

    % Υπολογισμός του βήματος κβαντισμού
    delta = (max_value - min_value) / L;

    % Υπολογισμός των κέντρων των περιοχών κβαντισμού
    centers = linspace(min_value + delta / 2, max_value - delta / 2, L);

    % Κατανομή κάθε δείγματος στην κατάλληλη περιοχή κβαντισμού
    xq = ceil((x - min_value) / delta);
    xq(xq == 0) = 1;  % Για τιμές ακριβώς στο min_value
    xq(xq > L) = L;   % Για τιμές ακριβώς στο max_value

    % Επιστροφή του διανύσματος εξόδου xq και των κέντρων κβαντισμού
end

% Παράδειγμα χρήσης της συνάρτησης
[y, fs] = audioread('speech.wav'); % Φόρτωση του σήματος
min_value = -1;                    % Ελάχιστη τιμή του σήματος
max_value = 1;                     % Μέγιστη τιμή του σήματος
N = 4;                             % Αριθμός bits

% Κανονικοποίηση του σήματος
y = y / max(abs(y));

% Κλήση του ομοιόμορφου κβαντιστή
[xq, centers] = my_quantizer(y, N, min_value, max_value);

% Ανακατασκευή του κβαντισμένου σήματος
reconstructed_signal = centers(xq);

% Αναπαραγωγή του αρχικού και του κβαντισμένου σήματος
sound(y, fs);
pause(length(y) / fs);
sound(reconstructed_signal, fs);
```

run_quantizer.m

```
% Φόρτωση του σήματος από το αρχείο
[y, fs] = audioread('speech.wav');

% Ορισμός παραμέτρων του κβαντιστή
min_value = -1;                    % Ελάχιστη τιμή του σήματος
max_value = 1;                     % Μέγιστη τιμή του σήματος
N = 16;                             % Αριθμός bits
```

```
% Κανονικοποίηση του σήματος
y = y / max(abs(y));


% Εκτέλεση του ομοιόμορφου κβαντιστή
[xq, centers] = my_quantizer(y, N, min_value, max_value);

% Ανακατασκευή του κβαντισμένου σήματος
reconstructed_signal = centers(xq);

% Εμφάνιση αποτελεσμάτων κβαντισμού
fprintf('Αποτελέσματα Κβαντισμού:\n');
fprintf('Αριθμός Bits: %d\n', N);
fprintf('Επίπεδα Κβαντισμού: %d\n', 2^N);
fprintf('Βήμα Κβαντισμού (Delta): %.4f\n', (max_value - min_value) / (2^N));

fprintf('Πρώτα 10 δείγματα του αρχικού σήματος:\n');
disp(y(1:10));

fprintf('Πρώτα 10 δείκτες κβαντισμού:\n');
disp(xq(1:10));

fprintf('Πρώτα 10 τιμές ανακατασκευασμένου σήματος:\n');
disp(reconstructed_signal(1:10));
```

lloyd_max_quantizer.m

```
% Φόρτωση του σήματος από το αρχείο
[y, fs] = audioread('speech.wav');

% Ορισμός παραμέτρων του κβαντιστή
min_value = -1;                    % Ελάχιστη τιμή του σήματος
max_value = 1;                     % Μέγιστη τιμή του σήματος
N = 16;                             % Αριθμός bits

% Κανονικοποίηση του σήματος
y = y / max(abs(y));

% Εκτέλεση του ομοιόμορφου κβαντιστή
[xq, centers] = my_quantizer(y, N, min_value, max_value);

% Ανακατασκευή του κβαντισμένου σήματος
reconstructed_signal = centers(xq);

% Εμφάνιση αποτελεσμάτων κβαντισμού
fprintf('Αποτελέσματα Κβαντισμού:\n');
fprintf('Αριθμός Bits: %d\n', N);
fprintf('Επίπεδα Κβαντισμού: %d\n', 2^N);
fprintf('Βήμα Κβαντισμού (Delta): %.4f\n', (max_value - min_value) / (2^N));

fprintf('Πρώτα 10 δείγματα του αρχικού σήματος:\n');
disp(y(1:10));

fprintf('Πρώτα 10 δείκτες κβαντισμού:\n');
disp(xq(1:10));

fprintf('Πρώτα 10 τιμές ανακατασκευασμένου σήματος:\n');
disp(reconstructed_signal(1:10));
```

run_pcm_quantizer.m

```
% Φόρτωση του σήματος από το αρχείο
[y, fs] = audioread('speech.wav');

% Ορισμός παραμέτρων του κβαντιστή
min_value = -1;                    % Ελάχιστη τιμή του σήματος
max_value = 1;                     % Μέγιστη τιμή του σήματος
N = 16;                             % Αριθμός bits

% Κανονικοποίηση του σήματος
y = y / max(abs(y));

% Εκτέλεση του ομοιόμορφου κβαντιστή
[xq, centers] = my_quantizer(y, N, min_value, max_value);

% Ανακατασκευή του κβαντισμένου σήματος
reconstructed_signal = centers(xq);

% Εμφάνιση αποτελεσμάτων κβαντισμού
fprintf('Αποτελέσματα Κβαντισμού:\n');
fprintf('Αριθμός Bits: %d\n', N);
fprintf('Επίπεδα Κβαντισμού: %d\n', 2^N);
fprintf('Βήμα Κβαντισμού (Delta): %.4f\n', (max_value - min_value) / (2^N));

fprintf('Πρώτα 10 δείγματα του αρχικού σήματος:\n');
disp(y(1:10));

fprintf('Πρώτα 10 δείκτες κβαντισμού:\n');
disp(xq(1:10));

fprintf('Πρώτα 10 τιμές ανακατασκευασμένου σήματος:\n');
disp(reconstructed_signal(1:10));
```

sqnr_lloyd_max_plot.m

```
% Φόρτωση του αρχείου ήχου
[y, fs] = audioread('speech.wav');
y = y / max(abs(y));               % Κανονικοποίηση του σήματος στο [-1, 1]

% Παράμετροι
min_value = -1;                    % Ελάχιστη τιμή
max_value = 1;                     % Μέγιστη τιμή
N_bits = [2, 4, 8];                % Bits κβαντισμού
max_iters = [10, 50, 100, 200];    % Αριθμός επαναλήψεων

% Προετοιμασία για το γράφημα
figure;
hold on;
line_styles = {'-', '--', '-.', ':'}; % Τύποι γραμμών για διαφορετικές επαναλήψεις

% Υπολογισμός και σχεδίαση SQNR για διαφορετικά N
SQNR_results_b = zeros(length(N_bits), length(max_iters)); % Αποθήκευση αποτελεσμάτων για σχήμα (b)
SQNR_results_a = zeros(1, length(N_bits));                 % Αποθήκευση αποτελεσμάτων για σχήμα (a)

% Υπολογισμός για σχήμα (a) - Ομοιόμορφος Κβαντιστής
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers = linspace(min_value + delta/2, max_value - delta/2, L);
```

```
    % Κβάντιση
    xq = min(max(round((y - min_value) / delta) + 1, 1), L);
    reconstructed_signal = centers(xq);

    % Υπολογισμός SQNR
    noise = y(:) - reconstructed_signal(:); % Θόρυβος
    signal_power = mean(y(:).^2);          % Ισχύς σήματος
    noise_power = mean(noise.^2);          % Ισχύς θορύβου
    SQNR = 10 * log10(signal_power / noise_power); % SQNR σε dB

    % Αποθήκευση αποτελεσμάτων
    SQNR_results_a(n_idx) = SQNR;
end

% Υπολογισμός για σχήμα (b) - Lloyd-Max Κβαντιστής
for k_idx = 1:length(max_iters)
    K_max = max_iters(k_idx);
    for n_idx = 1:length(N_bits)
        N = N_bits(n_idx);
        L = 2^N; % Επίπεδα κβαντισμού

        % Αρχικοποίηση Lloyd-Max
        centers = linspace(min_value, max_value, L); % Ισομερώς κατανεμημένα κέντρα
        thresholds = zeros(1, L + 1);
        thresholds(1) = -Inf;
        thresholds(end) = Inf;

        % Lloyd-Max επαναληπτικός αλγόριθμος
        for iter = 1:K_max
            % Υπολογισμός ορίων βάσει κέντρων
            for i = 2:L
                thresholds(i) = (centers(i-1) + centers(i)) / 2;
            end

            % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
            xq = zeros(size(y));
            for i = 1:L
                indices = y > thresholds(i) & y <= thresholds(i+1);
                xq(indices) = i;
            end

            % Αναπροσαρμογή των κέντρων
            new_centers = zeros(1, L);
            for i = 1:L
                region = y(y > thresholds(i) & y <= thresholds(i+1));
                if ~isempty(region)
                    new_centers(i) = mean(region);
                else
                    new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
                end
            end

            % Έλεγχος για σύγκλιση
            if max(abs(new_centers - centers)) < 1e-10
                break;
            end
            centers = new_centers;
        end

        % Ανακατασκευή του κβαντισμένου σήματος
        reconstructed_signal = centers(xq);
```

```
        % Υπολογισμός SQNR
        noise = y(:) - reconstructed_signal(:); % Θόρυβος
        signal_power = mean(y(:).^2);          % Ισχύς σήματος
        noise_power = mean(noise.^2);          % Ισχύς θορύβου
        SQNR = 10 * log10(signal_power / noise_power); % SQNR σε dB

        % Αποθήκευση του SQNR
        SQNR_results_b(n_idx, k_idx) = SQNR;
    end

    % Σχεδίαση καμπύλης για την τρέχουσα τιμή K_max
    plot(N_bits, SQNR_results_b(:, k_idx), ...
         'LineStyle', line_styles{k_idx}, ...
         'LineWidth', 1.5, ...
         'DisplayName', sprintf('K_{max} = %d (Μη Ομοιόμορφος)', K_max));
end

% Σχεδίαση αποτελεσμάτων για σχήμα (a)
plot(N_bits, SQNR_results_a, '-o', 'LineWidth', 2, 'DisplayName', 'Ομοιόμορφος');

% Προσθήκη λεπτομερειών στο γράφημα
xlabel('Αριθμός Bits (N)');
ylabel('SQNR (dB)');
title('Σύγκριση SQNR: Ομοιόμορφος vs Μη Ομοιόμορφος Κβαντιστής');
legend('show', 'Location', 'southeast'); % Εμφάνιση λεζάντας
grid on;
hold off;
```

compare_waveforms_quantizers.m

```
% Φόρτωση του αρχικού αρχείου ήχου
[y, fs] = audioread('speech.wav');
y = y / max(abs(y));               % Κανονικοποίηση του σήματος στο [-1, 1]

% Παράμετροι
min_value = -1;                    % Ελάχιστη τιμή
max_value = 1;                     % Μέγιστη τιμή
N_bits = [2, 4, 8];                % Bits κβαντισμού
K_max = 100;                       % Επαναλήψεις για τον Lloyd-Max
epsilon = 1e-6;                    % Όριο σύγκλισης για τον Lloyd-Max
t = (0:length(y)-1) / fs;          % Χρονικός άξονας

% Υπολογισμός και σχεδίαση για κάθε N
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού

    %% Ομοιόμορφος Κβαντιστής
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers_uniform = linspace(min_value + delta/2, max_value - delta/2, L); % Κέντρα
    xq_uniform = ceil((y - min_value) / delta);
    xq_uniform(xq_uniform == 0) = 1;    % Αν κάποιο δείγμα είναι στο min_value
    xq_uniform(xq_uniform > L) = L;     % Αν κάποιο δείγμα είναι στο max_value
    reconstructed_uniform = centers_uniform(xq_uniform); % Ανακατασκευή

    %% Lloyd-Max Κβαντιστής
    centers = linspace(min_value, max_value, L); % Αρχικά κέντρα
    thresholds = zeros(1, L + 1);
    thresholds(1) = -Inf;
    thresholds(end) = Inf;
```

```
% Lloyd-Max επαναληπτικός αλγόριθμος
    for iter = 1:K_max
        % Υπολογισμός ορίων βάσει κέντρων
        for i = 2:L
            thresholds(i) = (centers(i-1) + centers(i)) / 2;
        end

        % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
        xq = zeros(size(y));
        for i = 1:L
            indices = y > thresholds(i) & y <= thresholds(i+1);
            xq(indices) = i;
        end

        % Αναπροσαρμογή των κέντρων
        new_centers = zeros(1, L);
        for i = 1:L
            region = y(y > thresholds(i) & y <= thresholds(i+1));
            if ~isempty(region)
                new_centers(i) = mean(region);
            else
                new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
            end
        end

        % Έλεγχος για σύγκλιση
        if max(abs(new_centers - centers)) < epsilon
            break;
        end
        centers = new_centers;
    end

    % Ανακατασκευή του σήματος
    reconstructed_lloyd_max = centers(xq);

    %% Σχεδίαση Κυματομορφών
    figure; % Άνοιγμα νέου παραθύρου
    plot(t, y, 'k', 'LineWidth', 1.5, 'DisplayName', 'Αρχικό Σήμα'); % Αρχικό
    hold on;
    plot(t, reconstructed_uniform, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Ομοιόμορφος Κβαντιστής'); %
Ομοιόμορφος
    plot(t, reconstructed_lloyd_max, 'b-.', 'LineWidth', 1.5, 'DisplayName', 'Lloyd-Max Κβαντιστής'); %
Lloyd-Max
    xlabel('Χρόνος (s)');
    ylabel('Πλάτος');
    title(['Σύγκριση Κυματομορφών για N = ', num2str(N)]);
    legend('show');
    grid on;
    hold off;

    %% Αναπαραγωγή Σημάτων
    disp(['Αναπαραγωγή του αρχικού σήματος για N = ', num2str(N)]);
    sound(y, fs);
    pause(length(y) / fs + 1);

    disp(['Αναπαραγωγή του ομοιόμορφου σήματος για N = ', num2str(N)]);
    sound(reconstructed_uniform, fs);
    pause(length(y) / fs + 1);

    disp(['Αναπαραγωγή του Lloyd-Max σήματος για N = ', num2str(N)]);
    sound(reconstructed_lloyd_max, fs);
    pause(length(y) / fs + 1);
```

```
end
```

compare_mse_quantizers.m

```
% Φόρτωση του αρχικού αρχείου ήχου
[y, fs] = audioread('speech.wav');
y = y / max(abs(y));                % Κανονικοποίηση του σήματος στο [-1, 1]

% Παράμετροι
min_value = -1;                     % Ελάχιστη τιμή
max_value = 1;                      % Μέγιστη τιμή
N_bits = [2, 4, 8];                 % Bits κβαντισμού
K_max = 100;                        % Επαναλήψεις για τον Lloyd-Max
epsilon = 1e-6;                     % Όριο σύγκλισης για τον Lloyd-Max

% Αρχικοποίηση αποθήκευσης MSE
MSE_uniform = zeros(1, length(N_bits)); % MSE για τον ομοιόμορφο
MSE_lloyd_max = zeros(1, length(N_bits)); % MSE για τον Lloyd-Max

% Υπολογισμός MSE για κάθε τιμή N
for n_idx = 1:length(N_bits)
    N = N_bits(n_idx);
    L = 2^N; % Επίπεδα κβαντισμού

    %% Ομοιόμορφος Κβαντιστής
    delta = (max_value - min_value) / L; % Βήμα κβαντισμού
    centers_uniform = linspace(min_value + delta/2, max_value - delta/2, L); % Κέντρα
    xq_uniform = ceil((y - min_value) / delta);
    xq_uniform(xq_uniform == 0) = 1;    % Αν κάποιο δείγμα είναι στο min_value
    xq_uniform(xq_uniform > L) = L;     % Αν κάποιο δείγμα είναι στο max_value
    reconstructed_uniform = centers_uniform(xq_uniform); % Ανακατασκευή

    % Υπολογισμός MSE για τον ομοιόμορφο
    error_uniform = y(:) - reconstructed_uniform(:);
    MSE_uniform(n_idx) = mean(error_uniform.^2);

    %% Lloyd-Max Κβαντιστής
    centers = linspace(min_value, max_value, L); % Αρχικά κέντρα
    thresholds = zeros(1, L + 1);
    thresholds(1) = -Inf;
    thresholds(end) = Inf;

    % Lloyd-Max επαναληπτικός αλγόριθμος
    for iter = 1:K_max
        % Υπολογισμός ορίων βάσει κέντρων
        for i = 2:L
            thresholds(i) = (centers(i-1) + centers(i)) / 2;
        end

        % Αντιστοίχιση τιμών του σήματος στα επίπεδα κβαντισμού
        xq = zeros(size(y));
        for i = 1:L
            indices = y > thresholds(i) & y <= thresholds(i+1);
            xq(indices) = i;
        end

        % Αναπροσαρμογή των κέντρων
        new_centers = zeros(1, L);
        for i = 1:L
            region = y(y > thresholds(i) & y <= thresholds(i+1));
            if ~isempty(region)
```

```
                new_centers(i) = mean(region);
            else
                new_centers(i) = centers(i); % Διατήρηση αν δεν υπάρχουν δείγματα
            end
        end

        % Έλεγχος για σύγκλιση
        if max(abs(new_centers - centers)) < epsilon
            break;
        end
        centers = new_centers;
    end

    % Ανακατασκευή του σήματος
    reconstructed_lloyd_max = centers(xq);

    % Υπολογισμός MSE για τον Lloyd-Max
    error_lloyd_max = y(:) - reconstructed_lloyd_max(:);
    MSE_lloyd_max(n_idx) = mean(error_lloyd_max.^2);
end

%% Απεικόνιση MSE για τους δύο κβαντιστές
figure;
plot(N_bits, MSE_uniform, '-o', 'LineWidth', 1.5, 'DisplayName', 'Ομοιόμορφος Κβαντιστής');
hold on;
plot(N_bits, MSE_lloyd_max, '-s', 'LineWidth', 1.5, 'DisplayName', 'Lloyd-Max Κβαντιστής');
xlabel('Αριθμός Bits (N)');
ylabel('MSE');
title('Σύγκριση MSE Ομοιόμορφου και Lloyd-Max');
legend('show', 'Location', 'northeast'); % Εμφάνιση λεζάντας
grid on;
hold off;
```

dpcm_coding.m

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
N = 8; % Αριθμός bits του κβαντιστή
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής
quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

% Αρχικοποίηση
num_samples = length(t); % Χρήση της μεταβλητής t
quantized_signal = zeros(num_samples, 1);
reconstructed_signal = zeros(num_samples, 1);
error_signal = zeros(num_samples, 1);

% Συντελεστές φίλτρου πρόβλεψης (πριν τον κβαντισμό)
predictor_coeffs_original = [0.9];

% Παράμετροι του ομοιόμορφου κβαντιστή
min_value_coeffs = -2; % Ελάχιστη τιμή δυναμικής περιοχής
max_value_coeffs = 2;  % Μέγιστη τιμή δυναμικής περιοχής
delta_coeffs = (max_value_coeffs - min_value_coeffs) / (2^8); % Βήμα κβαντισμού
```

```
% Κβαντισμός των συντελεστών πρόβλεψης
predictor_coeffs = max(min(round(predictor_coeffs_original / delta_coeffs) * delta_coeffs,

max_value_coeffs), min_value_coeffs);

% Τάξη φίλτρου πρόβλεψης
predictor_order = length(predictor_coeffs);


%% Συνάρτηση για την πρόβλεψη
predict = @(coeffs, signal, n, order) ...
    sum(coeffs .* flip(signal(n - order + 1:n))); % Υπολογισμός πρόβλεψης

%% Συνάρτηση για κβαντισμό
quantize = @(value, delta, min_value, max_value) ...
    max(min(round(value / delta) * delta, max_value), min_value);

%% Κωδικοποίηση DPCM
for n = 1:num_samples
    if n <= predictor_order
        prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
    else
        % Υπολογισμός πρόβλεψης με βάση προηγούμενα δείγματα
        prediction = predict(predictor_coeffs, reconstructed_signal, n - 1, predictor_order);
    end

    % Υπολογισμός σφάλματος πρόβλεψης
    error_signal(n) = t(n) - prediction;

    % Κβαντισμός σφάλματος
    quantized_error = quantize(error_signal(n), delta, min_value, max_value);

    % Αποθήκευση κβαντισμένου σφάλματος
    quantized_signal(n) = quantized_error;

    % Ανακατασκευή του τρέχοντος δείγματος
    reconstructed_signal(n) = prediction + quantized_error;
end

%% Αποκωδικοποίηση DPCM
decoded_signal = zeros(num_samples, 1);
for n = 1:num_samples
    if n <= predictor_order
        prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
    else
        % Υπολογισμός πρόβλεψης με βάση τα προηγούμενα αποκωδικοποιημένα δείγματα
        prediction = predict(predictor_coeffs, decoded_signal, n - 1, predictor_order);
    end

    % Ανακατασκευή δείγματος
    decoded_signal(n) = prediction + quantized_signal(n);
end

%% Απεικόνιση Αποτελεσμάτων
figure;
subplot(3, 1, 1);
plot(t, 'b');
title('Αρχικό Σήμα');
xlabel('Δείγματα');
ylabel('Τιμή');

subplot(3, 1, 2);
```

```
plot(reconstructed_signal, 'r');
title('Ανακατασκευασμένο Σήμα (DPCM Encoder)');
xlabel('Δείγματα');
ylabel('Τιμή');


subplot(3, 1, 3);
plot(decoded_signal, 'g');
title('Αποκωδικοποιημένο Σήμα (DPCM Decoder)');
xlabel('Δείγματα');
ylabel('Τιμή');
```

dpcm_error_analysis.m

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής

% Επιλογή τιμών του p
p_values = [5, 7]; % Δύο διαφορετικές τιμές του p

% Αρχικοποίηση
num_samples = length(t);

% Σχεδίαση για κάθε τιμή του p
for p = p_values
    % Δημιουργία φίλτρου πρόβλεψης με p συντελεστές
    predictor_coeffs = ones(1, p) / p; % Απλό μέσο όρο p προηγούμενων τιμών
    predictor_order = length(predictor_coeffs);

    for N = 1:3 % Για κάθε αριθμό bits
        % Υπολογισμός παραμέτρων κβαντιστή
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

        % Αρχικοποίηση σημάτων
        quantized_signal = zeros(num_samples, 1);
        reconstructed_signal = zeros(num_samples, 1);
        error_signal = zeros(num_samples, 1);

        % Υπολογισμός σφάλματος πρόβλεψης
        for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης (χρήση προηγούμενων δειγμάτων)
                prediction = sum(predictor_coeffs .* t(n - predictor_order:n - 1)');
            end

            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal(n) = t(n) - prediction;

            % Κβαντισμός σφάλματος
            quantized_error = max(min(round(error_signal(n) / delta) * delta, max_value), min_value);
            quantized_signal(n) = quantized_error;
        end
```

```
        % Δημιουργία ξεχωριστού γραφήματος
        figure;
        plot(t, 'b', 'DisplayName', 'Αρχικό Σήμα');
        hold on;
        plot(error_signal, 'r', 'DisplayName', 'Σφάλμα Πρόβλεψης');
        title(sprintf('Αρχικό Σήμα και Σφάλμα Πρόβλεψης (p = %d, N = %d)', p, N));

       xlabel('Δείγματα');


       ylabel('Τιμή');
        legend;
        hold off;
    end
end
```

dpcm_mse_analysis.m

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής
p_values = [5, 10]; % Τιμές του p
N_values = [1, 2, 3]; % Τιμές των bits του κβαντιστή

% Αρχικοποίηση αποτελεσμάτων
mse_results = zeros(length(p_values), length(N_values)); % Μέσο τετραγωνικό σφάλμα για κάθε συνδυασμό

% Υπολογισμός MSE για κάθε συνδυασμό p και N
for p_idx = 1:length(p_values)
    p = p_values(p_idx); % Τρέχουσα τιμή του p

    % Δημιουργία συντελεστών προβλέπτη (απλός μέσος όρος p τιμών)
    predictor_coeffs = ones(1, p) / p; % Ομοιόμορφα κατανεμημένοι συντελεστές
    predictor_order = length(predictor_coeffs);
    fprintf('Συντελεστές προβλέπτη για p = %d: %s\n', p, mat2str(predictor_coeffs));

    for N_idx = 1:length(N_values)
        N = N_values(N_idx); % Τρέχουσα τιμή των bits
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

        % Υπολογισμός σφάλματος πρόβλεψης
        num_samples = length(t);
        error_signal = zeros(num_samples, 1);

        for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης
                prediction = sum(predictor_coeffs .* t(n - predictor_order:n - 1)');
            end

            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal(n) = t(n) - prediction;

            % Κβαντισμός σφάλματος
            error_signal(n) = max(min(round(error_signal(n) / delta) * delta, max_value), min_value);
```

```
      end

        % Υπολογισμός MSE
        mse_results(p_idx, N_idx) = mean(error_signal.^2);
    end
end




% Σχεδίαση αποτελεσμάτων
figure;
for p_idx = 1:length(p_values)
    p = p_values(p_idx);
    plot(N_values, mse_results(p_idx, :), '-o', 'DisplayName', sprintf('p = %d', p));
    hold on;
end
xlabel('Αριθμός bits N');
ylabel('Μέσο Τετραγωνικό Σφάλμα (MSE)');
title('MSE Πρόβλεψης ως προς τον Αριθμό Bits και την Τάξη Προβλέπτη');
legend('show');
grid on;
```

dpcm_reconstruction_analysis.m

```
% Φόρτωση των δεδομένων
load source.mat; % Τα δεδομένα αποθηκεύονται στη μεταβλητή "t"

% Ορισμός παραμέτρων
min_value = -3.5; % Ελάχιστη τιμή δυναμικής περιοχής
max_value = 3.5; % Μέγιστη τιμή δυναμικής περιοχής
p_values = [5, 10]; % Τιμές του p
N_values = [1, 2, 3]; % Τιμές των bits του κβαντιστή

% Αρχικοποίηση
num_samples = length(t);

% Υπολογισμός και σχεδίαση για κάθε συνδυασμό p και N
for p_idx = 1:length(p_values)
    p = p_values(p_idx); % Τρέχουσα τιμή του p

    % Δημιουργία συντελεστών προβλέπτη (απλός μέσος όρος p τιμών)
    predictor_coeffs = ones(1, p) / p; % Ομοιόμορφα κατανεμημένοι συντελεστές
    predictor_order = length(predictor_coeffs);

    for N_idx = 1:length(N_values)
        N = N_values(N_idx); % Τρέχουσα τιμή των bits
        quant_levels = 2^N; % Αριθμός επιπέδων κβαντιστή
        delta = (max_value - min_value) / quant_levels; % Βήμα κβαντιστή

        % Υπολογισμός ανακατασκευασμένου σήματος
        quantized_signal = zeros(num_samples, 1);
        reconstructed_signal = zeros(num_samples, 1);

        for n = 1:num_samples
            if n <= predictor_order
                prediction = 0; % Μηδενική πρόβλεψη για τα πρώτα δείγματα
            else
                % Υπολογισμός πρόβλεψης
                prediction = sum(predictor_coeffs .* reconstructed_signal(n - predictor_order:n - 1)');
            end
```

```
            % Υπολογισμός σφάλματος πρόβλεψης
            error_signal = t(n) - prediction;

            % Κβαντισμός σφάλματος
            quantized_error = max(min(round(error_signal / delta) * delta, max_value), min_value);
            quantized_signal(n) = quantized_error;


           % Ανακατασκευή σήματος
            reconstructed_signal(n) = prediction + quantized_error;

end

        % Σχεδίαση του αρχικού και του ανακατασκευασμένου σήματος
        figure;
        plot(t, 'b', 'DisplayName', 'Αρχικό Σήμα');
        hold on;
        plot(reconstructed_signal, 'r', 'DisplayName', 'Ανακατασκευασμένο Σήμα');
        title(sprintf('Αρχικό και Ανακατασκευασμένο Σήμα (p = %d, N = %d)', p, N));
        xlabel('Δείγματα');
        ylabel('Τιμή');
        legend;
        grid on;
        hold off;
    end
end
```

ΜΕΡΟΣ Β

MPAM.m

```
% Παράμετροι
M = 4; % Αριθμός συμβόλων (M-PAM)
N = 1000; % Αριθμός συμβόλων που θα σταλούν
fc = 1000; % Συχνότητα φέρουσας (Hz)
fs = 10000; % Συχνότητα δειγματοληψίας (Hz)
T = 1/100; % Διάρκεια συμβόλου (s)
t = 0:1/fs:T-1/fs; % Χρονικός άξονας για κάθε σύμβολο

% Αστερισμός M-PAM
A = 2 * (1:M) - M - 1;

% Τυχαία συμβολική ακολουθία
symbols = A(randi([1, M], 1, N));

% Διαμόρφωση
s_t = zeros(1, length(t) * N);
for i = 1:N
    s_t((i-1)*length(t)+1:i*length(t)) = symbols(i) * cos(2*pi*fc*t);
end

% Προσθήκη θορύβου (χειροκίνητα)
SNR = 20; % Σήμα προς Θόρυβο σε dB
signal_power = mean(s_t.^2); % Υπολογισμός ισχύος σήματος
noise_power = signal_power / (10^(SNR/10)); % Υπολογισμός ισχύος θορύβου
noise = sqrt(noise_power) * randn(size(s_t)); % Δημιουργία Gaussian θορύβου
r_t = s_t + noise; % Προσθήκη θορύβου στο σήμα

% Αποδιαμόρφωση
r = zeros(1, N);
```

```
for i = 1:N
    segment = r_t((i-1)*length(t)+1:i*length(t));
    r(i) = sum(segment .* cos(2*pi*fc*t)) / length(t);
end

% Φωρατής
detected_symbols = zeros(1, N);
for i = 1:N
    [~, idx] = min(abs(A - r(i)));  % Χρησιμοποιούμε το A αντί του Am
    detected_symbols(i) = A(idx);  % Ανίχνευση του συμβόλου
end


% Υπολογισμός σφαλμάτων
errors = sum(detected_symbols ~= symbols);

% Απεικόνιση
figure;
subplot(3, 1, 1);
stem(symbols(1:50), 'b', 'DisplayName', 'Original Symbols');
title('Original Symbols');
xlabel('Symbol Index');
ylabel('Amplitude');
grid on;

subplot(3, 1, 2);
stem(r(1:50), 'r', 'DisplayName', 'Received Signal');
title('Received Signal');
xlabel('Symbol Index');
ylabel('Amplitude');
grid on;

subplot(3, 1, 3);
stem(detected_symbols(1:50), 'g', 'DisplayName', 'Detected Symbols');
title('Detected Symbols');
xlabel('Symbol Index');
ylabel('Amplitude');
grid on;

disp(['Total Errors: ', num2str(errors)]);
```

mpam_prediction_error.m

```
% Παράμετροι συστήματος
Lb = 100000; % Αρχικό πλήθος bits
SNR_dB = 0:2:20; % Εύρος τιμών SNR
M_values = [2, 8]; % Τιμές M

% Αρχικοποίηση αποτελεσμάτων
results = struct();



for M = M_values
    k = log2(M); % Bits per symbol

    % Προσαρμογή του πλήθους των bits ώστε να είναι πολλαπλάσιο του k
    Lb_adjusted = floor(Lb / k) * k; % Πολλαπλάσιο του k
    bits = randi([0, 1], Lb_adjusted, 1); % Τυχαία ακολουθία bits
```

```
    % Βήμα 2: Αντιστοίχιση bits σε σύμβολα
    num_symbols = Lb_adjusted / k; % Αριθμός συμβόλων
    bit_groups = reshape(bits, k, []); % Ομαδοποίηση σε ομάδες k bits
    symbols = zeros(1, num_symbols);

    for i = 1:num_symbols
        symbols(i) = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0)); % Υπολογισμός συμβόλου
    end

    % Βήμα 3: Διαμόρφωση M-PAM
    Es = 1; % Ενέργεια ανά σύμβολο

    A = sqrt(3 * Es / (M^2 - 1)); % Πλάτος
    pam_symbols = A * (2 * symbols - M + 1); % Διαμόρφωση PAM

    % Αρχικοποίηση BER και SER
    BER = zeros(size(SNR_dB));
    SER = zeros(size(SNR_dB));

    for idx = 1:length(SNR_dB)
        % Βήμα 4: Προσθήκη θορύβου
        noise_variance = Es / (10^(SNR_dB(idx)/10)); % Διασπορά θορύβου
        noise = sqrt(noise_variance) * randn(size(pam_symbols)); % Θόρυβος AWGN
        received_signal = pam_symbols + noise; % Σήμα με θόρυβο

        % Βήμα 5: Απόφαση συμβόλων
        detected_symbols = round((received_signal / A + M - 1) / 2); % Απόφαση
        detected_symbols = max(min(detected_symbols, M-1), 0); % Περιορισμός εύρους

        % Αντίστροφη αντιστοίχιση συμβόλων σε bits
        detected_bits = zeros(Lb_adjusted, 1);
        for j = 1:num_symbols
            dec_value = detected_symbols(j);
            for b = 1:k
                detected_bits((j-1)*k + b) = mod(floor(dec_value / 2^(k-b)), 2);
            end
        end

        % Υπολογισμός BER και SER
        BER(idx) = sum(bits ~= detected_bits) / Lb_adjusted;
        SER(idx) = sum(symbols ~= detected_symbols) / num_symbols;
    end

    % Αποθήκευση αποτελεσμάτων
    results(M).SNR_dB = SNR_dB;
    results(M).BER = BER;
    results(M).SER = SER;
end

% Βήμα 6: Γραφήματα
figure;
hold on;
for M = M_values
    semilogy(results(M).SNR_dB, results(M).BER, '-o', 'DisplayName', sprintf('BER for M=%d', M));


end
xlabel('SNR (dB)');
ylabel('Error Rate');
legend;
grid on;
title('BER για M-PAM με απλή κωδικοποίηση');
```

```
% Ερώτημα 2: M = 8 με κωδικοποίηση Gray
M = 8;
k = log2(M);
Lb_adjusted = floor(Lb / k) * k;
bits = randi([0, 1], Lb_adjusted, 1);
num_symbols = Lb_adjusted / k;

bit_groups = reshape(bits, k, []);
symbols_gray = zeros(1, num_symbols);

% Αντιστοίχιση Gray
gray_map = bitxor((0:M-1)', floor((0:M-1)'/2));
for i = 1:num_symbols
    binary_value = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0));

    symbols_gray(i) = find(gray_map == binary_value) - 1;
end

% Διαμόρφωση με Gray
pam_symbols_gray = A * (2 * symbols_gray - M + 1);
BER_gray = zeros(size(SNR_dB));

for idx = 1:length(SNR_dB)
    noise_variance = Es / (10^(SNR_dB(idx)/10));
    noise = sqrt(noise_variance) * randn(size(pam_symbols_gray));
    received_signal = pam_symbols_gray + noise;

    % Απόφαση Gray
    detected_symbols = round((received_signal / A + M - 1) / 2);
    detected_symbols = max(min(detected_symbols, M-1), 0);

    % Αντιστοίχιση Gray
    detected_bits = zeros(Lb_adjusted, 1);
    for j = 1:num_symbols
        gray_value = gray_map(detected_symbols(j) + 1);
        for b = 1:k
            detected_bits((j-1)*k + b) = mod(floor(gray_value / 2^(k-b)), 2);
        end
    end

    % Υπολογισμός BER Gray
    BER_gray(idx) = sum(bits ~= detected_bits) / Lb_adjusted;
end

% Προσθήκη Gray στο γράφημα
semilogy(SNR_dB, BER_gray, '-x', 'DisplayName', 'BER for M=8 (Gray)');
xlabel('SNR (dB)');
ylabel('Error Rate');
legend;
grid on;
title('BER για M-PAM με M=2, M=8 (απλή και Gray κωδικοποίηση)');

% Εκτύπωση αποτελεσμάτων για M=2
disp('Αποτελέσματα για M=2:');
disp(table(SNR_dB', results(2).BER', 'VariableNames', {'SNR_dB', 'BER_M2'}));

% Εκτύπωση αποτελεσμάτων για M=8 με απλή κωδικοποίηση
disp('Αποτελέσματα για M=8 (Απλή Κωδικοποίηση):');
disp(table(SNR_dB', results(8).BER', 'VariableNames', {'SNR_dB', 'BER_M8_Simple'}));
```

```
% Εκτύπωση αποτελεσμάτων για M=8 με κωδικοποίηση Gray
disp('Αποτελέσματα για M=8 (Κωδικοποίηση Gray):');
disp(table(SNR_dB', BER_gray', 'VariableNames', {'SNR_dB', 'BER_M8_Gray'}));
```

ser_mpam_simulation.m

```
% Παράμετροι συστήματος
Lb = 100000; % Αρχικό πλήθος bits

SNR_dB = 0:2:20; % Εύρος τιμών SNR
M_values = [2, 8]; % Τιμές M

% Αρχικοποίηση αποτελεσμάτων
results = struct();

for M = M_values
    k = log2(M); % Bits per symbol

    % Προσαρμογή του πλήθους των bits ώστε να είναι πολλαπλάσιο του k
    Lb_adjusted = floor(Lb / k) * k; % Πολλαπλάσιο του k
    bits = randi([0, 1], Lb_adjusted, 1); % Τυχαία ακολουθία bits

    % Βήμα 2: Αντιστοίχιση bits σε σύμβολα
    num_symbols = Lb_adjusted / k; % Αριθμός συμβόλων
    bit_groups = reshape(bits, k, []); % Ομαδοποίηση σε ομάδες k bits
    symbols = zeros(1, num_symbols);
    for i = 1:num_symbols
        symbols(i) = sum(bit_groups(:, i)' .* 2.^(k-1:-1:0)); % Υπολογισμός συμβόλου
    end

    % Βήμα 3: Διαμόρφωση M-PAM
    Es = 1; % Ενέργεια ανά σύμβολο
    A = sqrt(3 * Es / (M^2 - 1)); % Πλάτος
    pam_symbols = A * (2 * symbols - M + 1); % Διαμόρφωση PAM

    % Αρχικοποίηση SER
    SER = zeros(size(SNR_dB));

    for idx = 1:length(SNR_dB)
        % Βήμα 4: Προσθήκη θορύβου
        noise_variance = Es / (10^(SNR_dB(idx)/10)); % Διασπορά θορύβου
        noise = sqrt(noise_variance) * randn(size(pam_symbols)); % Θόρυβος AWGN
        received_signal = pam_symbols + noise; % Σήμα με θόρυβο

        % Βήμα 5: Απόφαση συμβόλων
        detected_symbols = round((received_signal / A + M - 1) / 2); % Απόφαση
        detected_symbols = max(min(detected_symbols, M-1), 0); % Περιορισμός εύρους

        % Υπολογισμός SER
        SER(idx) = sum(symbols ~= detected_symbols) / num_symbols;
    end

    % Αποθήκευση αποτελεσμάτων
    results(M).SNR_dB = SNR_dB;
    results(M).SER = SER;



 % Εκτύπωση πιθανοτήτων σφάλματος συμβόλου
    disp(['Αποτελέσματα για M = ', num2str(M), ':']);
```

```
    disp(table(SNR_dB', SER', 'VariableNames', {'SNR_dB', 'SER'}));
end

% Γραφήματα SER
figure;
hold on;
for M = M_values
    semilogy(results(M).SNR_dB, results(M).SER, '-o', 'DisplayName', sprintf('SER for M=%d', M));
end
xlabel('SNR (dB)');
ylabel('Symbol Error Rate (SER)');
legend;
grid on;
title('SER για M-PAM με απλή κωδικοποίηση');
```