ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

Ακαδημαϊκό Έτος 2024-2025

1η Εργαστηριακή Άσκηση

Φοιτητής: Μπερτσεκάς Παρασκευάς-Σωτήριος

ΑΜ: 1093445

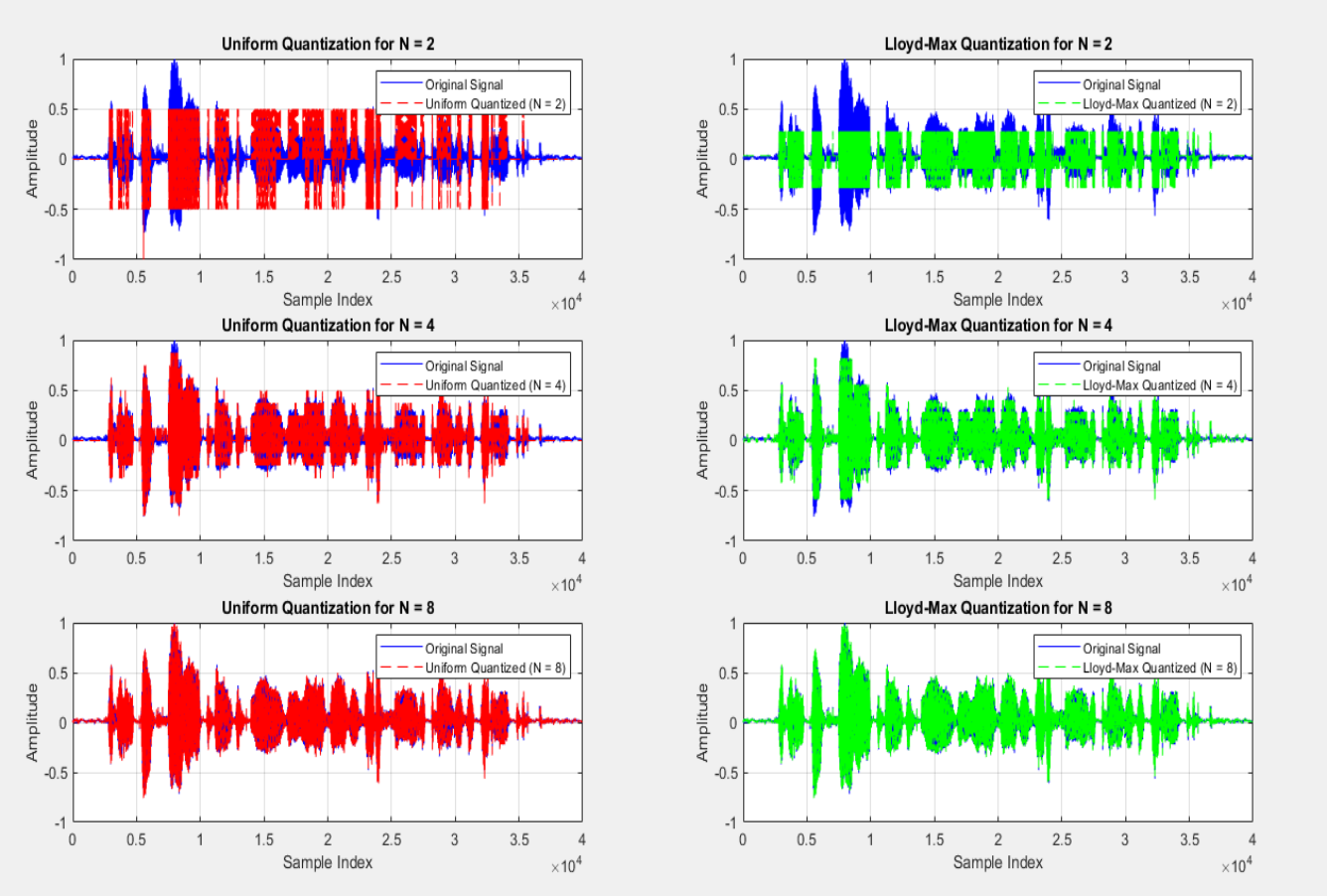
Μέρος Α

Κωδικοποίηση Πηγής με PCM

Ερωτήματα Μέρους Α1

1. Υλοποίηση PCM

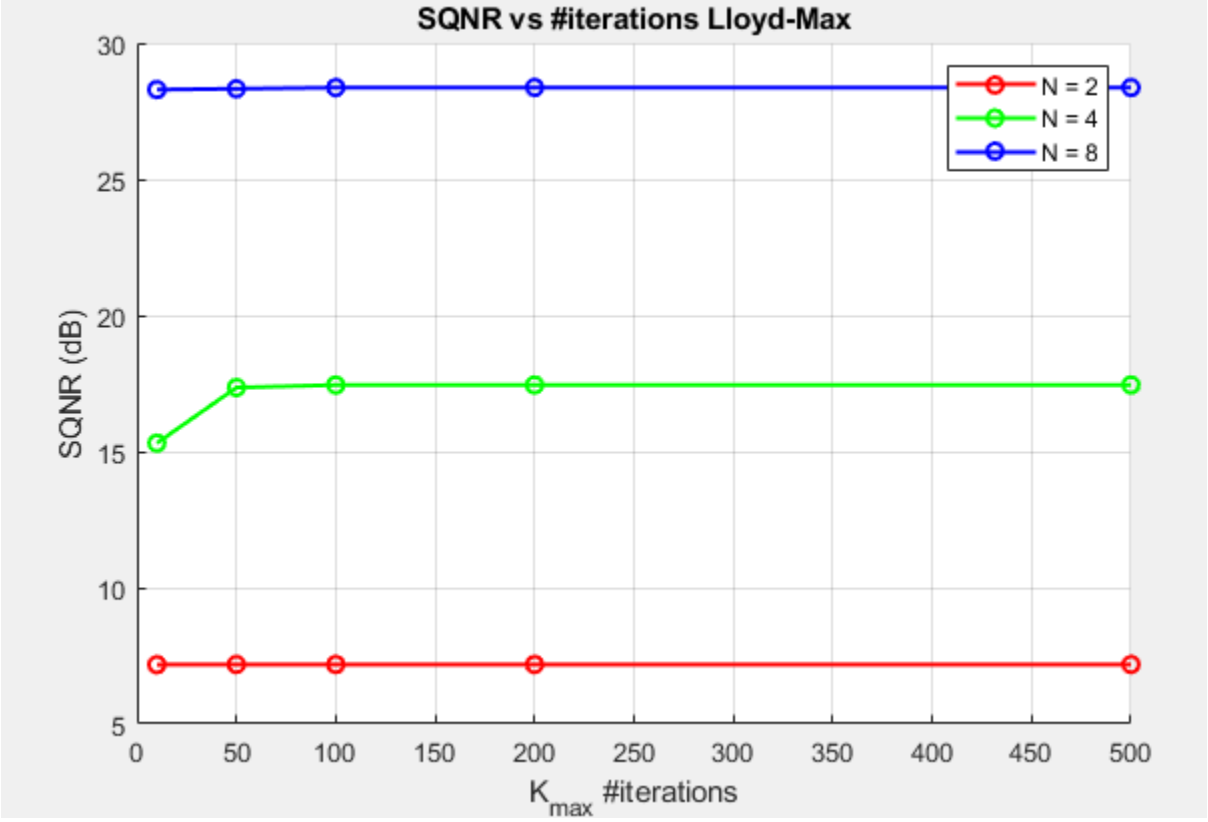
Ο κώδικας για τις 2 υλοποίησεις και για την σύγκριση τους παρατίθεται στο τέλος της αναφοράς. Παρ' όλα αυτά, εδώ έχουμε μερικές διαφορές ανάμεσα στους δύο, με την σύγκριση όπως ζητείται στα επόμενα ερωτήματα.



Παρατηρώντας προσεκτικά, διαπιστώνεται ότι ο ομοιόμορφος κβαντιστής χρησιμοποιεί σταθερό βήμα κβάντισης σε όλο το φάσμα του σήματος. Αυτό έχει ως συνέπεια τα επίπεδα κβάντισης να είναι ομοιόμορφα κατανεμημένα. Σε περιοχές του σήματος με μικρές διακυμάνσεις, το σφάλμα αναπαράστασης είναι μεγαλύτερο, καθώς η σταθερή ακρίβεια κβάντισης δεν προσαρμόζεται στις τοπικές ιδιαιτερότητες του σήματος. Αντίθετα, ο κβαντιστής Lloyd-Max ρυθμίζει τα επίπεδα κβάντισης με βάση την κατανομή του σήματος, με αποτέλεσμα περισσότερα επίπεδα να τοποθετούνται σε περιοχές υψηλής πυκνότητας και λιγότερα σε περιοχές χαμηλής πυκνότητας. Έτσι, μειώνεται το σφάλμα κβάντισης στις περιοχές με έντονες μεταβολές του σήματος, βελτιώνοντας την ακρίβεια της αναπαράστασης.

2. Κωδικοποίηση πηγής ήχου για min_value = -1, max_value = 1 και N = 2, 4 και 8 bits. αξιολόγηση PCM με βάση τα ακόλουθα:

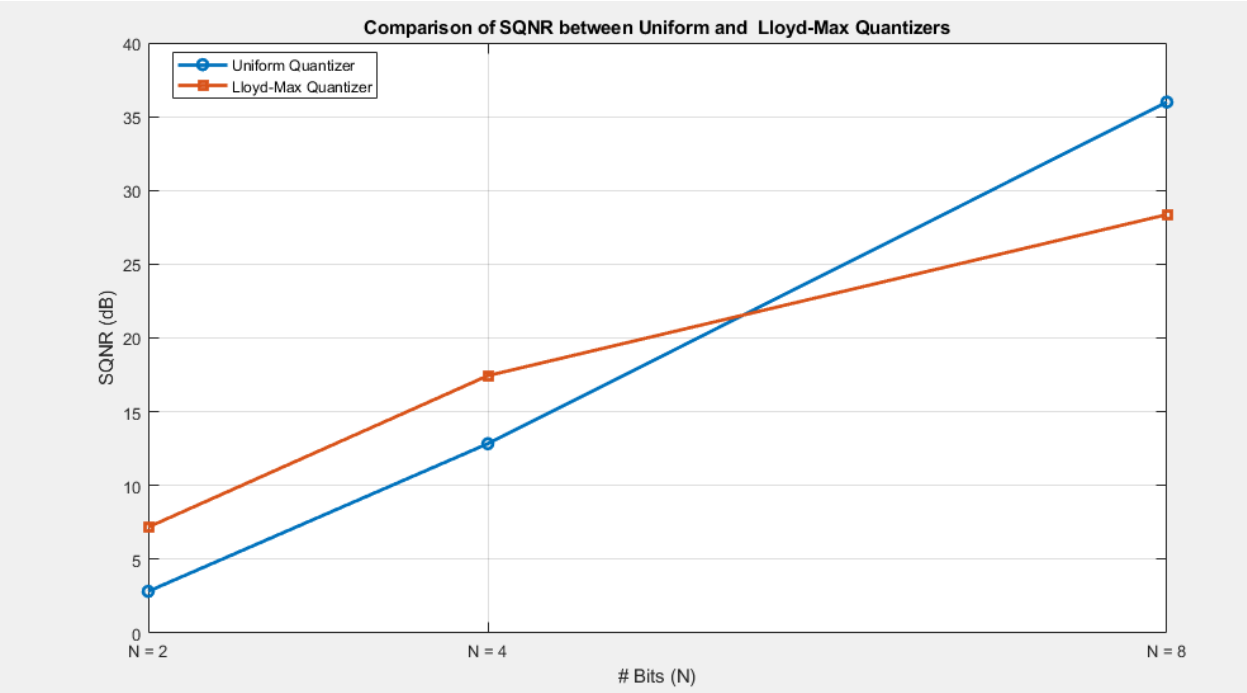
i. Σχεδιασμός μεταβολής του SQNR σε σχέση με τον αριθμό επαναλήψεων του αλγορίθμου για τον μη ομοιόμορφο κβαντιστή.



Όπως φαίνεται στο γράφημα, το SQNR (Signal to Quantization Noise Ratio) αυξάνεται σταδιακά με την αύξηση του αριθμού επαναλήψεων Kmax του αλγορίθμου Lloyd-Max, έως ότου φτάσει σε μια μέγιστη και σταθερή τιμή. Για

παράδειγμα, όταν N=4, στις αρχικές επαναλήψεις παρατηρείται μια μικρή αλλά σταθερή βελτίωση του SQNR, που αποδίδεται στη σύγκλιση του αλγορίθμου προς τα βέλτιστα κέντρα κβάντισης. Μετά από έναν συγκεκριμένο αριθμό επαναλήψεων (π.χ., περίπου 50), το SQNR σταθεροποιείται, υποδεικνύοντας ότι ο αλγόριθμος έχει πλήρως συγκλίνει και επιπλέον επαναλήψεις δεν προσφέρουν περαιτέρω βελτίωση.

ii. Σύγκριση τιμής του SQNR μετά από K επαναλήψεις και απόδοση 2 κβαντιστών:



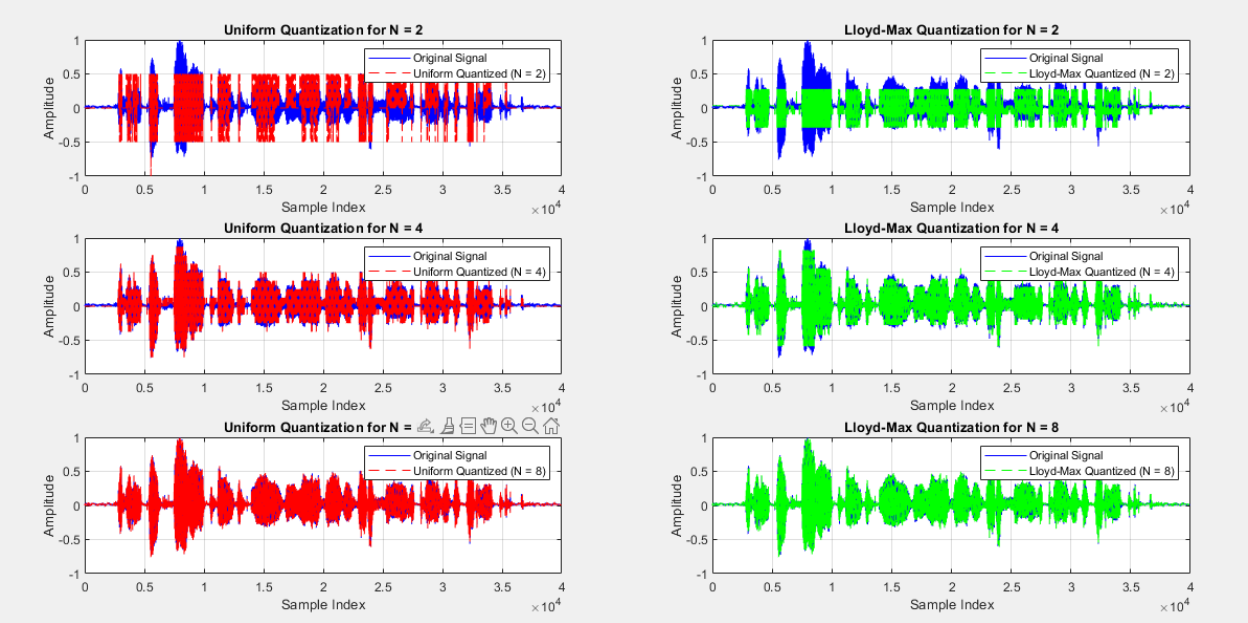
Results of SQNR comparison:

N (Bits)	SQNR Uniform (dB)	SQNR LloydMax (dB)
2	2.8287	7.1811
4	12.835	17.443
8	35.989	28.364

- Για N=2: ο Lloyd-Max κβαντιστής παρουσιάζει υψηλότερο SQNR (7.1811 dB) σε σχέση με τον ομοιόμορφο κβαντιστή (2.8287 dB). Αυτό δείχνει ότι ο Lloyd-Max χρησιμοποιεί πολύ καλύτερα τα διαθέσιμα επίπεδα κβάντισης προσαρμόζοντας τα κέντρα του στις πιθανότητες εμφάνισης των δειγμάτων.

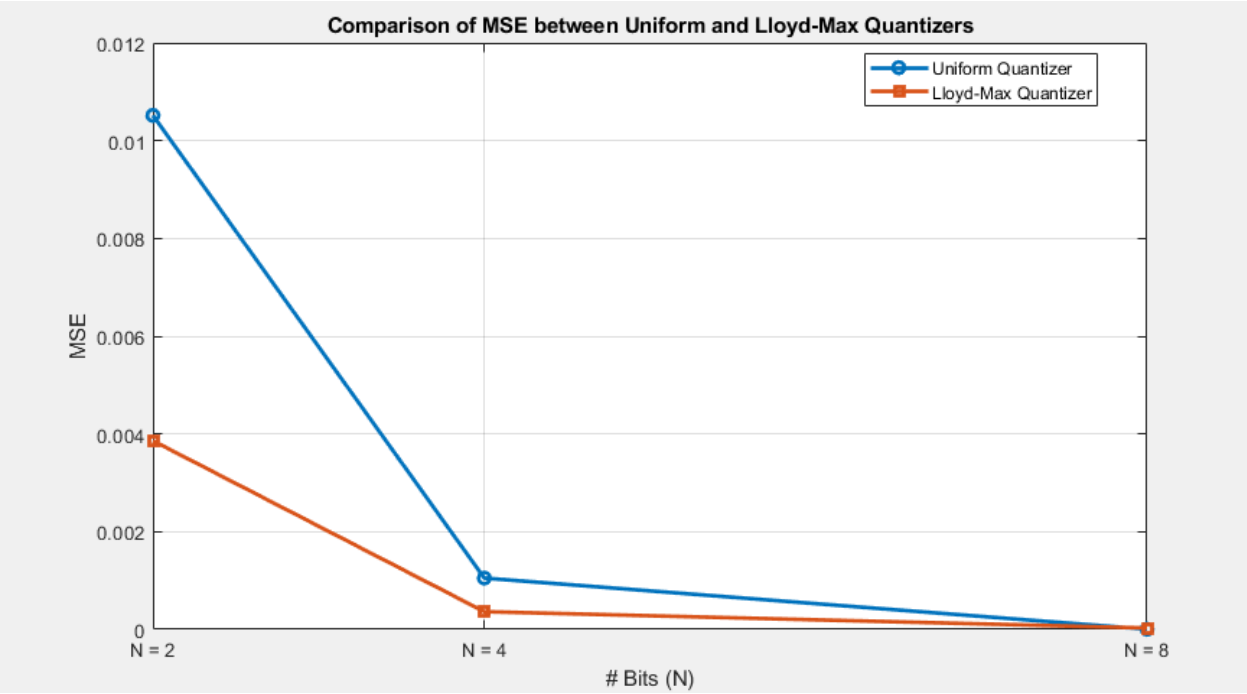
- Για N=4: ο Lloyd-Max παραμένει καλύτερος, με SQNR 17.443 dB σε σχέση με 12.835 dB του ομοιόμορφου κβαντιστή. Η διαφορά μεταξύ των 2 σαφώς μειώνεται, κάνοντας προφανές ότι με την αύξηση του N, ο ομοιόμορφος κβαντιστής πλησιάζει την απόδοση του Lloyd-Max.
- Για N=8: το SQNR του ομοιόμορφου κβαντιστή (35.989 dB) ξεπερνά το SQNR του Lloyd-Max (28.364 dB). Αυτό το αποτέλεσμα μπορεί να οφείλεται στη κατανομή των δειγμάτων (στατική), αφού ο ομοιόμορφος κβαντιστής κατανέμει ίσα τα επίπεδα του, εφόσον τον ωφελεί ο μεγάλος αριθμός bits.
- Από τα παραπάνω καταλαβαίνουμε ότι ο LloydMax ειναι καλύτερος για μικρές τιμές του N, όπου η προσαρμοστικότητα του στις πιθανότητες εμφάνισης των δειγμάτων έχει μεγαλύτερη σημασία, ενώ για μεγαλύτερες τιμές του N, η διαφορά μειώνεται, και σε κάποιες περιπτώσεις ο ομοιόμορφος κβαντιστής μπορεί να επιτύχει καλύτερο SQNR λόγω της ίσης κατανομής των επιπέδων του.

iii. Ακουστική αξιόλογη αποτελεσμάτων χρησιμοποιώντας την sound ():



Για κάθε τιμή του N, πραγματοποιήθηκε ακουστική αναπαραγωγή τόσο του κβαντισμένου σήματος με ομοιόμορφο κβαντιστή όσο και με τον κβαντιστή Lloyd-Max. Κατά την ακουστική σύγκριση, παρατηρήθηκε ότι ο ομοιόμορφος κβαντιστής διατηρεί ικανοποιητική ποιότητα ήχου, ιδιαίτερα για μεγαλύτερες τιμές του N, ωστόσο για μικρότερες τιμές εμφανίζονται σημαντικές παραμορφώσεις. Αντίθετα, ο κβαντιστής Lloyd-Max προσφέρει ανώτερη ποιότητα ήχου σε όλες τις τιμές του N, με τη βελτίωση να είναι πιο εμφανής στις μικρότερες τιμές, ενώ το τελικό σήμα για N=8 είναι σχεδόν ταυτόσημο με το αρχικό. Τα παραπάνω μπορούν να επιβεβαιωθούν εκτελώντας τον κώδικα που παρατίθεται στο τέλος της αναφοράς στην αντίστοιχη ενότητα.

iv. Σχολιασμός αποδοτικότητας βάσει Μέσου Τετραγωνικού Σφάλματος (MSE) σε σχέση με τις τιμές του N:



Results of MSE comparison:

N (Bits)	MSE Uniform	MSE LloydMax
2	0.010516	0.0038604
4	0.0010502	0.00036346
8	5.0791e-06	2.94e-05

Η αξιολόγηση της αποδοτικότητας της κωδικοποίησης PCM, βασισμένη στο Μέσο Τετραγωνικό Σφάλμα (MSE), έδειξε ότι ο μη ομοιόμορφος κβαντιστής Lloyd-Max υπερτερεί του ομοιόμορφου κβαντιστή για όλες τις τιμές του N. Συγκεκριμένα, για χαμηλές τιμές του N (π.χ., N=2), ο κβαντιστής Lloyd-Max επιτυγχάνει σημαντικά μικρότερο MSE σε σχέση με τον ομοιόμορφο κβαντιστή, καθώς προσαρμόζεται στη στατιστική κατανομή του σήματος και κατανέμει τα επίπεδα κβάντισης πιο αποδοτικά. Καθώς το N αυξάνεται (π.χ., N=8), και οι δύο κβαντιστές τείνουν να συγκλίνουν σε χαμηλά επίπεδα MSE, όμως ο Lloyd-Max συνεχίζει να παρουσιάζει μικρότερο σφάλμα, επιβεβαιώνοντας την υπεροχή του. Η γραφική απεικόνιση των αποτελεσμάτων αναδεικνύει αυτή τη σύγκριση, με τις καμπύλες MSE του Lloyd-Max να είναι συνεχώς κάτω από αυτές του ομοιόμορφου κβαντιστή, επισημαίνοντας την καλύτερη απόδοσή του. Συνεπώς, καταλήγουμε στο συμπέρασμα ότι ο Lloyd-Max κβαντιστής είναι πιο αποδοτικός, ιδιαίτερα για μικρές τιμές N, καθιστώντας τον κατάλληλο για εφαρμογές που απαιτούν χαμηλό σφάλμα κβάντισης.

Κωδικοποίηση Πηγής με DPCM

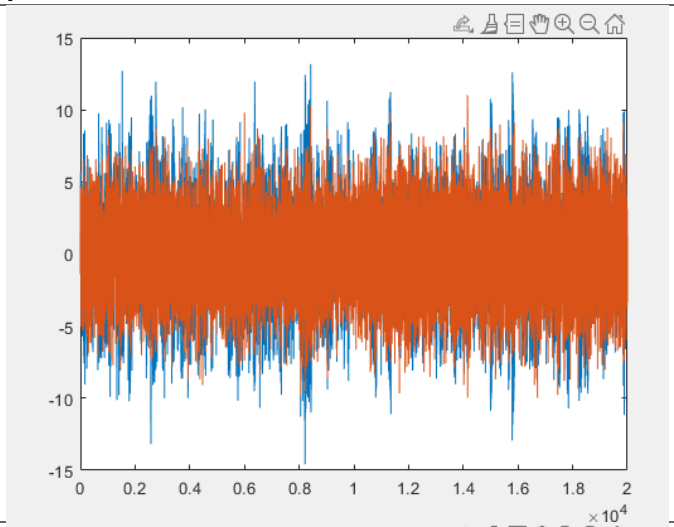
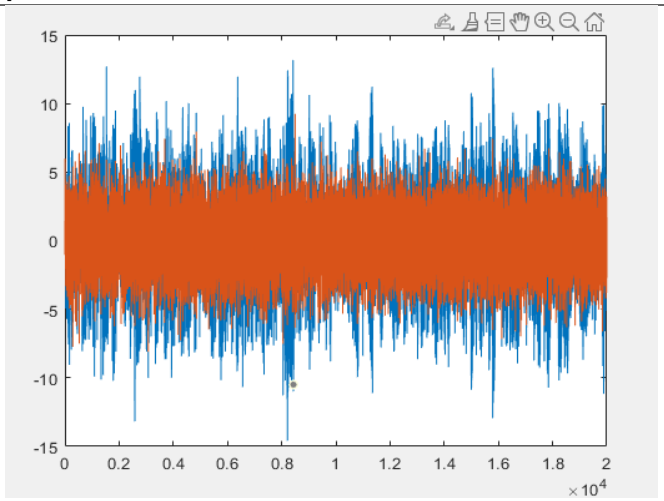
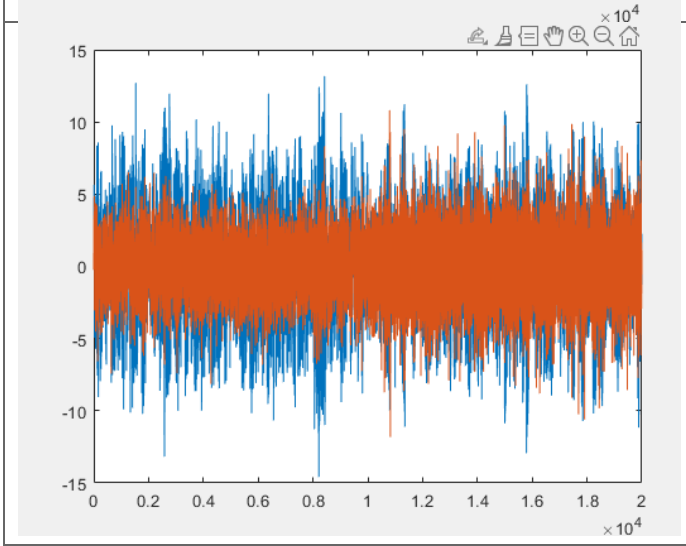
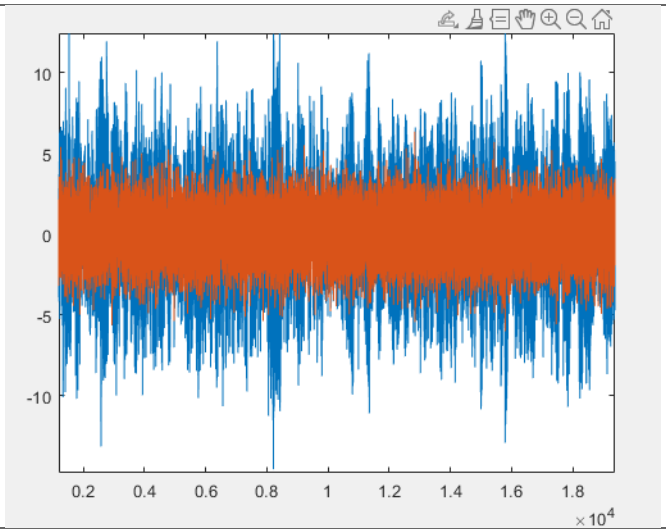
Ερωτήματα Μέρους Α2

1. Υλοποίηση συστήματος κωδικοποίησης/αποκωδικοποίησης DPCM:

Το σύστημα έχει υλοποιηθεί σύμφωνα με τις απαιτήσεις της εργασίας και ο κώδικας του βρίσκεται στο τέλος της αναφοράς.

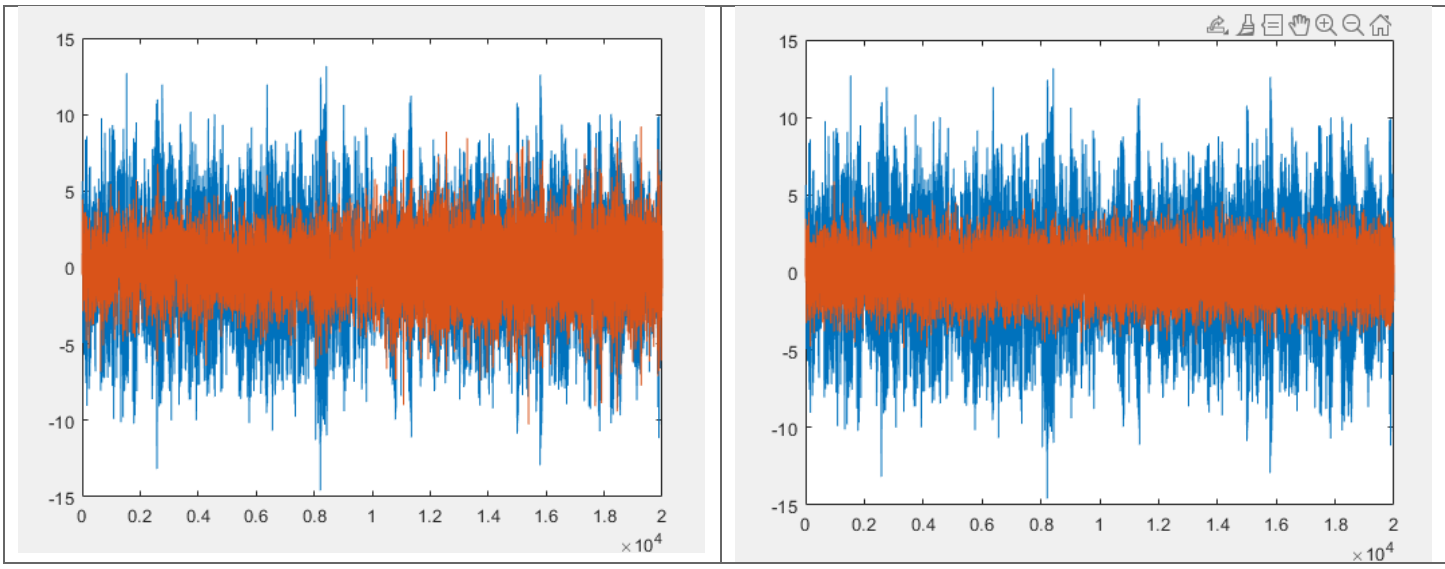
2. Σχεδίαση αρχικού σήματος και σφάλμα πρόβλεψης y στο ίδιο γράφημα και σχολιασμός αποτελεσμάτων.

Μας γίνεται ξεκάθαρο ότι όσο μεγαλύτερο το p, τόσο καλύτερη η πρόβλεψη και όσο μεγαλύτερο το N, τόσο μικρότερο σφάλμα εισάγει η κβάντιση. Και οι δύο παράμετροι, όσο ανεβαίνουν, μειώνουν το σφάλμα y.

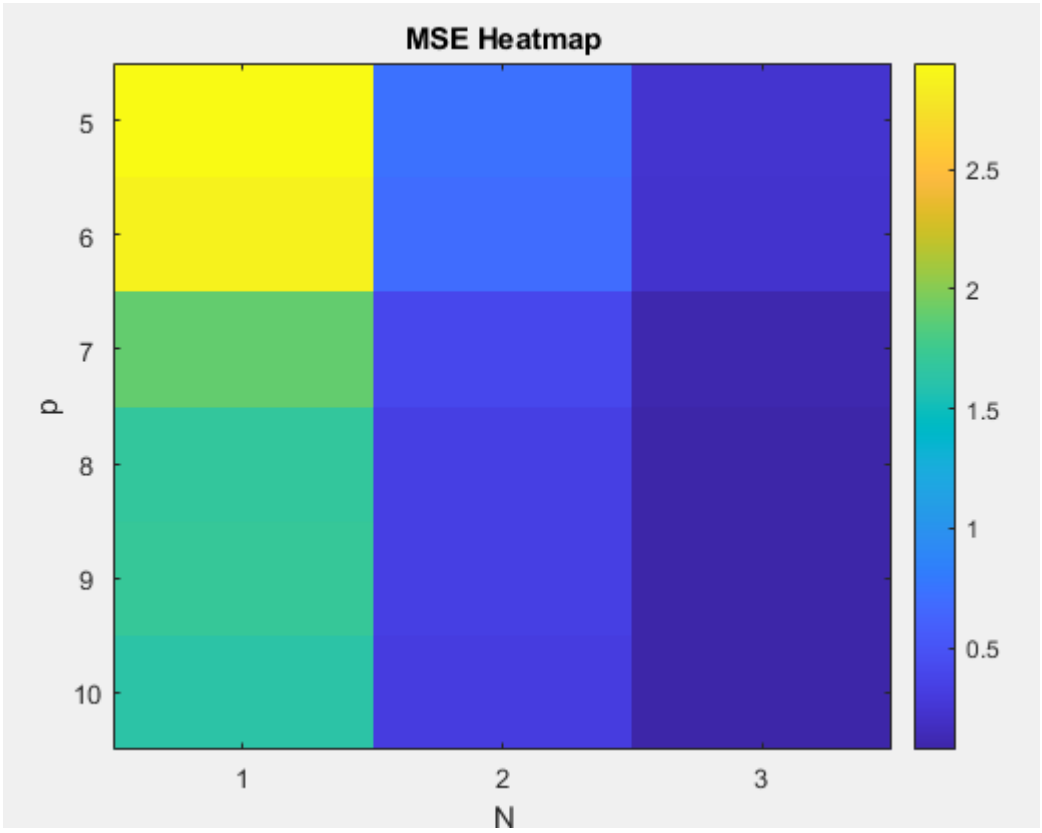
	p=5	**p=10**
N=1		
N=2		

Μπερτσεκάς Παρασκευάς-Σωτήριος AM:1093445

N=3



3. Αξιολόγηση απόδοσης με γράφημα στο οποίο απεικονίζεται το μέσο τετραγωνικό σφάλμα πρόβλεψης ως προς N, για διάφορες τιμές του p.

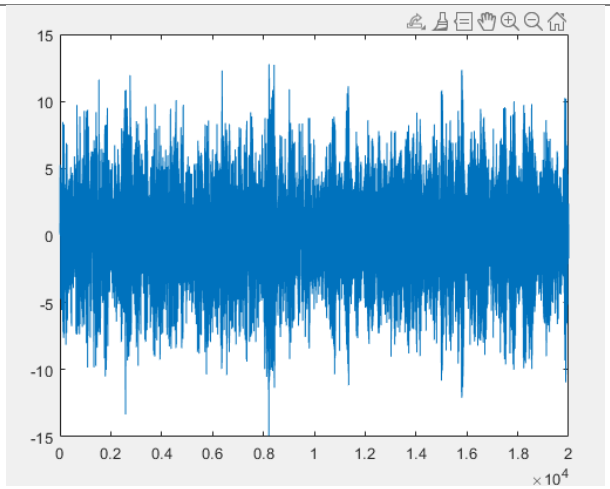


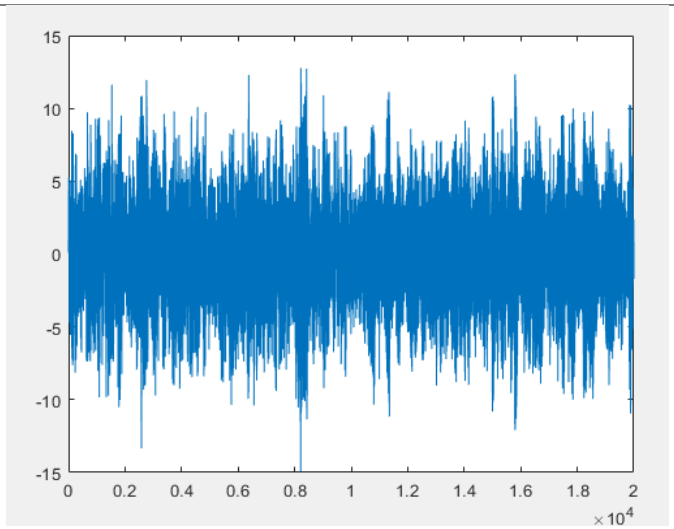
Στο γράφημα (heatmap) μπορούμε να αντιληφθούμε πως μεταβάλλεται το μέσο τετραγωνικό σφάλμα σε σχέση με τις παραμέτρους N και p. Όπως είχαμε ήδη υποψιαστεί από το προηγούμενο ερώτημα, υψηλότερες τιμές N και p, έχουν καλύτερο ανακατασκευασμένο σήμα (με μικρότερο MSE). Είναι επίσης σημαντικό να παρατηρήσουμε ότι από ένα σημείο και μετά το p δεν συνεισφέρει σημαντικά στην μείωση του MSE. Επίσης αξίζει να αναφερθούμε στο ότι η αύξηση του N οδηγεί σε δραματική μείωση του MSE.

4. Αρχικό σήμα και ανακατασκευασμένο σήμα στο δέκτη για p =5 και p =10.

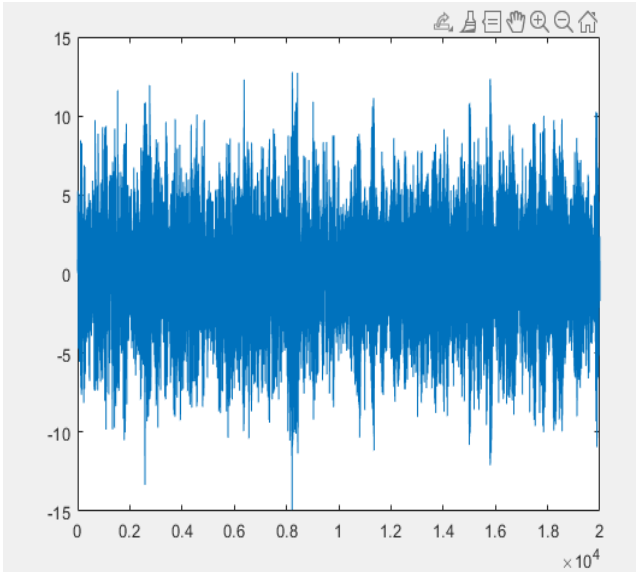
p=5 *p=10*

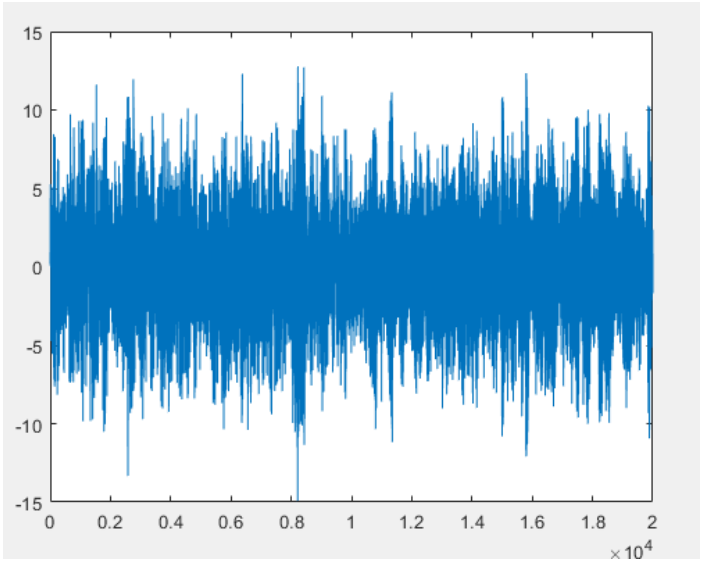
N=1



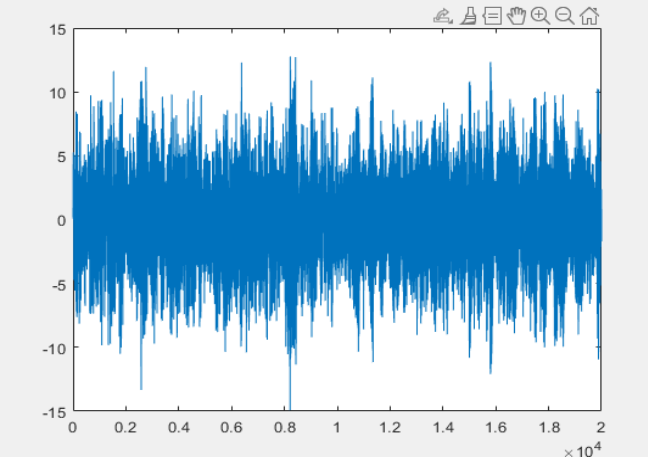


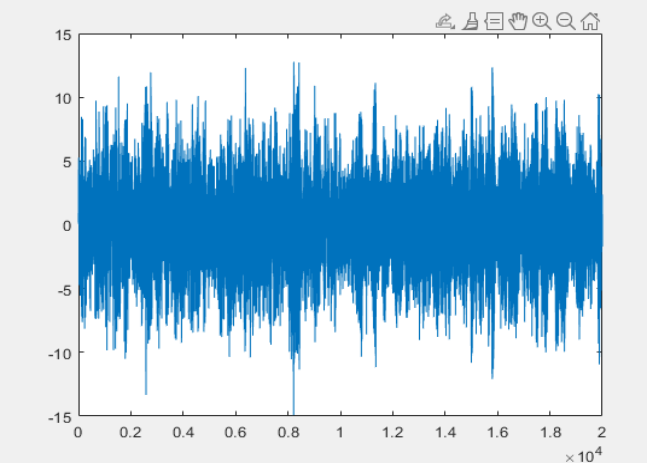
N=2



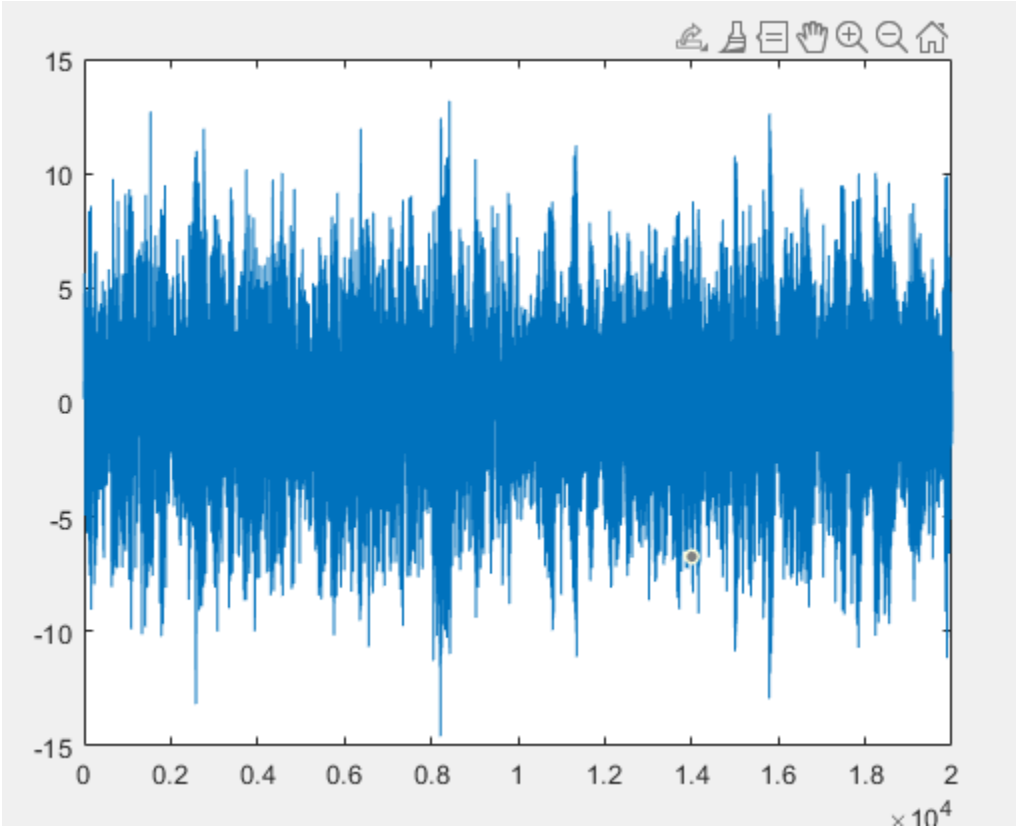


N=3





Αρχικό σήμα:



Σε σύγκριση και με τα ανακατασκευασμένα σήματα παρατηρούμε ότι όσο μεγαλώνει το N, η ανακατασκευή μας μπορεί να ακολουθήσει όλο και πιο απότομες μεταβολές του αρχικού σήματος και αυτό διότι, αν παρατηρήσουμε, όσο αυξάνεται το N, τόσο οι «κορυφές» της κυμματομορφής, πλησιάζουν τις κορυφές του αρχικού σήματος.

Μέρος Β

Μελέτη Απόδοσης Ομόδυνου Ζωνοπερατού Συστήματος MPAM

Ερωτήματα μέρους Β

1. Υλοποίηση Συστήματος M-PAM

Ο κώδικας (παρατήθεται και στο τέλος της αναφοράς) ο κώδικας υλοποιεί και συγκρίνει την απόδοση διαφορετικών συστημάτων M-PAM (Pulse Amplitude Modulation) σε σχέση με τον ρυθμό σφαλμάτων bit (BER) και τον ρυθμό σφαλμάτων συμβόλων (SER), με ή χωρίς τη χρήση Gray κωδικοποίησης για το 8-PAM. Ειδικότερα, ο κώδικας λειτουργεί ως εξής:

1. Αρχικοποίηση μεταβλητών

2. Βρόχος για κάθε επίπεδο διαμόρφωσης (M)

Για κάθε επίπεδο διαμόρφωσης (2 ή 8):

Υπολογισμός παραμέτρων

Τυχαία ακολουθία bits δημιουργείται και προσαρμόζεται ώστε το μήκος να είναι συμβατό με τα σύμβολα του M-PAM.

Χαρτογράφηση bits σε σύμβολα:

Τα bits ομαδοποιούνται και μετατρέπονται σε επίπεδα M-PAM χρησιμοποιώντας την απόσταση μεταξύ των επιπέδων.

Μεταφορά σήματος:

Δημιουργείται ένα παλμικό σήμα και διαμορφώνεται το σήμα φορέα με τα σύμβολα.

3. Προσομοίωση καναλιού AWGN

- Για κάθε τιμή SNR:
  - Προσθήκη θορύβου Gauss
  - Αποδιαμόρφωση και ανάκτηση συμβόλων
  - Υπολογισμός BER και SER

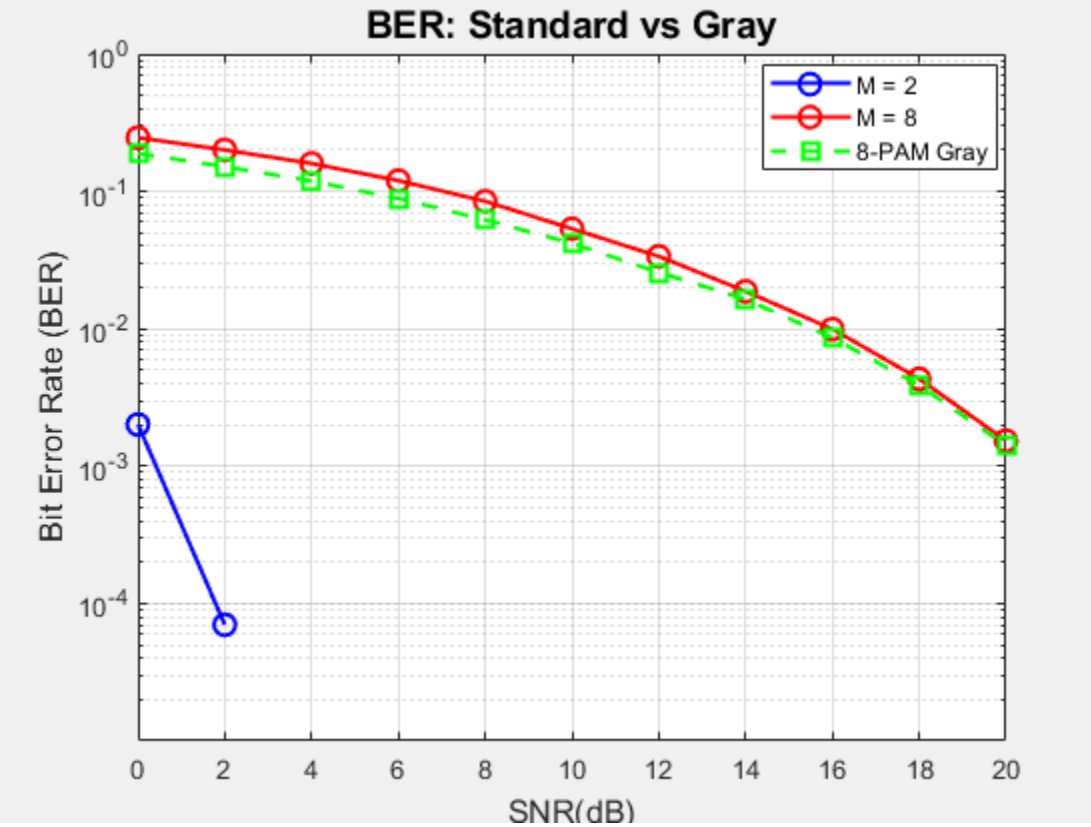
4. Gray Κωδικοποίηση για 8-PAM

- Ειδικά για το 8-PAM:
  - Gray Χαρτογράφηση
  - Προσομοίωση θορυβώδους καναλιού με Gray
  - Υπολογισμός Gray BER

5. Αποτελέσματα

- Τα αποτελέσματα εμφανίζονται ως πίνακας που περιλαμβάνει:
  - Το επίπεδο διαμόρφωσης (M).
  - Την τιμή του SNR.
  - Τα BER και SER για κάθε M.
  - Το BER Gray για το 8-PAM.
- Παράγει δύο plots:
  - Γράφημα BER
  - Γράφημα SER

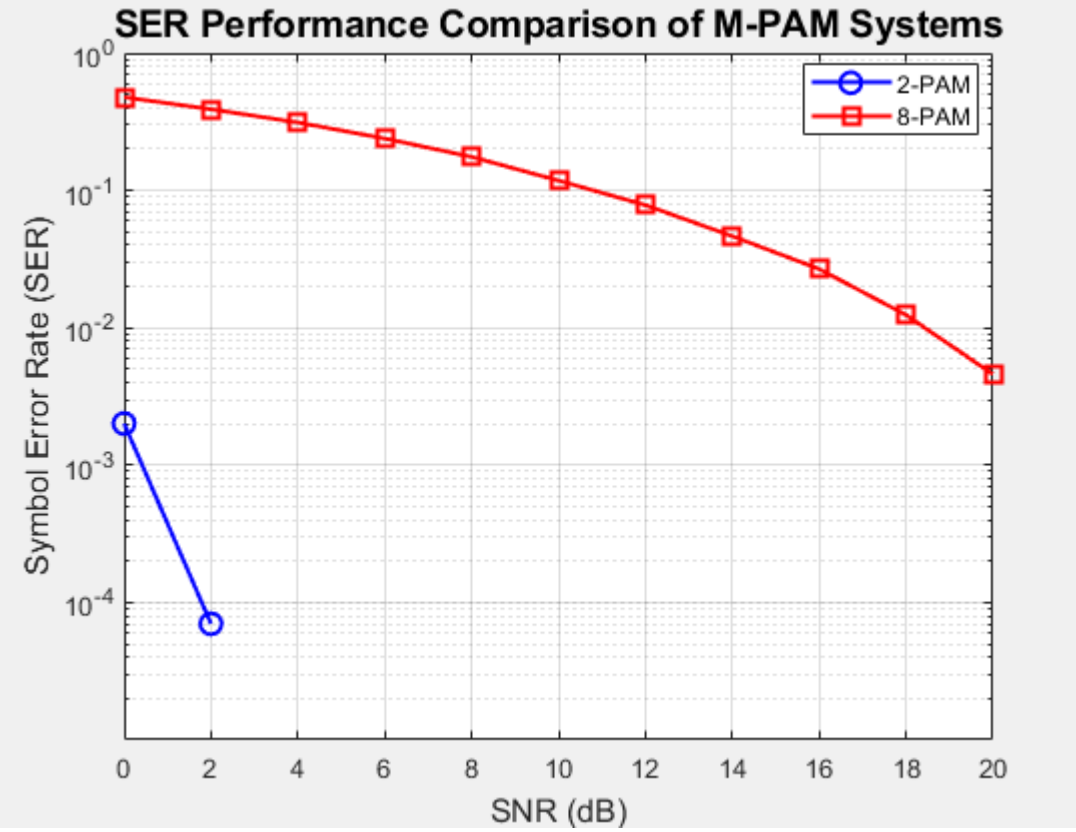
2. Διάγραμμα BER για απλή κωδικοποίηση και για κωδικοποίηση Gray



Τα αποτελέσματα στο γράφημα επιβεβαιώνουν τη θεωρία. Συγκεκριμένα, όσο αυξάνεται το SNR, το BER μειώνεται, κάτι που είναι αναμενόμενο, καθώς το SNR είναι ο λόγος της ισχύος του σήματος προς την ισχύ του θορύβου. Σε χαμηλές τιμές SNR, το σήμα επηρεάζεται περισσότερο από τον θόρυβο, προκαλώντας αλλοιώσεις και αυξάνοντας την πιθανότητα λανθασμένης ανίχνευσης bit. Αντίθετα, σε υψηλό SNR, το σήμα υπερισχύει του θορύβου, μειώνοντας τα σφάλματα. Έτσι, το BER μειώνεται καθώς αυξάνεται το SNR, όπως φαίνεται στο γράφημα. Όσον αφορά την τάξη του PAM (M), παρατηρούμε ότι το BER είναι μεγαλύτερο όταν η τάξη αυξάνεται. Αυτό συμβαίνει γιατί όταν M=2, υπάρχουν μόνο δύο επίπεδα πλάτους με μεγάλες αποστάσεις μεταξύ τους, ενώ όταν M=8, υπάρχουν περισσότερα επίπεδα με μικρότερες αποστάσεις. Οι μικρές αποστάσεις κάνουν τα

σύμβολα πιο ευάλωτα στον θόρυβο, οδηγώντας σε περισσότερα σφάλματα. Στο γράφημα φαίνεται ότι η PAM τάξης 2 φτάνει σε πολύ χαμηλό BER γρήγορα, από την τρίτη τιμή SNR, λόγω της μεγαλύτερης απόστασης μεταξύ των επιπέδων. Επιπλέον, η χρήση κωδικοποίησης Gray παρουσιάζει πλεονέκτημα. Με την κωδικοποίηση Gray, τα γειτονικά επίπεδα πλάτους διαφέρουν μόνο κατά 1 bit. Έτσι, αν ο θόρυβος μετατοπίσει το λαμβανόμενο σύμβολο σε γειτονικό επίπεδο, μόνο ένα bit θα είναι λανθασμένο. Σε αντίθεση, με την απλή κωδικοποίηση, περισσότερα bit μπορεί να επηρεαστούν. Επειδή ο θόρυβος έχει Gaussian κατανομή, οι μικρές μετατοπίσεις είναι πιο συχνές. Αυτό εξηγεί γιατί το BER της κωδικοποίησης Gray είναι μικρότερο και γιατί η καμπύλη της στο γράφημα βρίσκεται χαμηλότερα από την απλή κωδικοποίηση.

3. Διάγραμμα SER



Η καμπύλη του SER μειώνεται καθώς αυξάνεται το SNR, κάτι που είναι αναμενόμενο, καθώς η αύξηση του SNR μειώνει τον θόρυβο και, κατά συνέπεια, τα σφάλματα κατά την ανίχνευση μειώνονται επίσης. Αυτό έχει ως αποτέλεσμα η καμπύλη SER, που εκφράζει την πιθανότητα σφάλματος συμβόλου, να ακολουθεί φθίνουσα πορεία, όπως φαίνεται και στο γράφημα. Όσον αφορά την τάξη του PAM, όπως αναλύθηκε και προηγουμένως, η αύξηση της τάξης του PAM αυξάνει τον αριθμό των επιπέδων στα οποία αντιστοιχίζονται τα σύμβολα. Ως αποτέλεσμα, οι αποστάσεις μεταξύ αυτών των επιπέδων μειώνονται, καθιστώντας την αντιστοίχηση των συμβόλων στον δέκτη πιο ευάλωτη σε σφάλματα. Αυτή η συμπεριφορά απεικονίζεται στο γράφημα.

Επιπλέον, παρατηρούμε ότι για PAM δεύτερης τάξης, το SER φτάνει σε εξαιρετικά χαμηλές τιμές από την τρίτη αύξηση της τιμής του SNR. Αυτό το γεγονός καταδεικνύει ακριβώς τη συμπεριφορά που περιγράφηκε παραπάνω.

Κώδικας που χρησιμοποιήθηκε για όλα τα ερωτήματα:

Μέρος Α1

1.a.

```
[y, fs] = audioread('speech.wav');
min_value = -1;
max_value = 1;
y_norm = y / max(abs(y)); %normalisation
N = 4; % #bits
L = 2^N; % #levels kvantisis
delta = (max_value - min_value) / L; % vima kvantisis
centers = linspace(min_value + delta/2, max_value - delta/2, L);
xq = round((y_norm - min_value) / delta); % leveling
xq = max(min(xq, L - 1), 0); % exasfalisi oti ta levels einai anamesa [0, L-1]
quantized_signal = centers(xq + 1);
```

1.b. , 2.i.

```
[y, fs] = audioread('speech.wav');
signal = y / max(abs(y)); %normalisation
% Parameters
min_value = -1;
max_value = 1;
N_values = [2, 4, 8]; %  N (bits)
K_max_values = [10, 50, 100, 200, 500]; % K_max
epsilon = 1e-6; % threshold
SQNR_results = zeros(length(N_values), length(K_max_values));

for n_idx = 1:length(N_values)
 N = N_values(n_idx);
 L = 2^N; % #levels kvantisis
 delta = (max_value - min_value) / L; %vima kvantisis
 for k_idx = 1:length(K_max_values)
 K_max = K_max_values(k_idx);
 % Lloyd-Max
 centers = linspace(min_value + delta/2, max_value - delta/2, L);
 for k = 1:K_max
 boundaries = [min_value, (centers(1:end-1) + centers(2:end))/2, max_value];
%ipologismos boundaries
 xq_L_M = zeros(size(signal)); % kvantisi, samples se levels
 for i = 1:length(signal)
 for j = 1:L
 if signal(i) >= boundaries(j) && signal(i) < boundaries(j+1)
 xq_L_M(i) = centers(j);
 break;
 end
 end
```

```
 end
 new_centers = zeros(1, L);
 for j = 1:L
 indices = signal >= boundaries(j) & signal < boundaries(j+1);
 if sum(indices) > 0
 new_centers(j) = mean(signal(indices));
 else
 new_centers(j) = centers(j); % an den exoume nea deigmata kratame ta proigoumena
 end
 end
 if max(abs(new_centers - centers)) < epsilon %check to stop
 break;
 end
 centers = new_centers;
 end
 signal_power = mean(signal.^2);
 noise_power = mean((signal - xq_L_M).^2);
 SQNR_results(n_idx, k_idx) = 10 * log10(signal_power / noise_power); %sqnr se db
 end
end
%SQNR
figure;
hold on;
colors = ['r', 'g', 'b'];
for n_idx = 1:length(N_values)
 plot(K_max_values, SQNR_results(n_idx, :), '-o', 'Color', colors(n_idx),
'LineWidth', 1.5, ...
    'DisplayName', ['N = ', num2str(N_values(n_idx))]);
end
hold off;
title('SQNR vs #iterations Lloyd-Max');
xlabel('K_{max} #iterations');
ylabel('SQNR (dB)');
grid on;
legend('show');
```

2. ii., iii, iv

```
% Load the signal
[y, fs] = audioread('speech.wav');
signal = y / max(abs(y)); % normalisation
%Parameters
min_value = -1;
max_value = 1;
N_values = [2, 4, 8]; % N (bits)
K_max = 100; % max iters gia Lloyd-Max
epsilon = 1e-6; % threshold gia lloyd
% results kai plot counter gia ta plots
results = [];
results_mse = [];
plot_counter = 1;

for N = N_values
 L = 2^N; % Levels gia kvantisi
```

```
 delta = (max_value - min_value) / L; % vima kvantisis
 % uniform quantizer
 xq_uni = round((signal - min_value) / delta); % antistoixisi levels
 xq_uni = max(min(xq_uni, L - 1), 0); %exasfalisi levels [0, L-1]
 s_quantized_uni = min_value + xq_uni * delta; % kvantismenes times
 SQNR_uni = calculate_sqnr(signal, s_quantized_uni); %sqnr
 MSE_uni = mean((signal - s_quantized_uni).^2); %mse

 % Lloyd-Max Quantizer (non-uniform)
 centers = linspace(min_value + delta/2, max_value - delta/2, L);
 for k = 1:K_max
 boundaries = [min_value, (centers(1:end-1) + centers(2:end))/2, max_value];
%ipologismos boundaries
 xq_L_M = zeros(size(signal)); %deigmata se zones
 for i = 1:length(signal)
 for j = 1:L
 if signal(i) >= boundaries(j) && signal(i) < boundaries(j+1)
 xq_L_M(i) = centers(j);
 break;
 end
 end
 end
 new_centers = zeros(1, L);
 for j = 1:L
 indices = signal >= boundaries(j) & signal < boundaries(j+1);
 if sum(indices) > 0
 new_centers(j) = mean(signal(indices));
 else
 new_centers(j) = centers(j); % an den exoume nea deigmata kratame ta proigoumena
 end
 end
 if max(abs(new_centers - centers)) < epsilon %check to stop
 break;
 end
 centers = new_centers;
 end
 SQNR_L_M = calculate_sqnr(signal, xq_L_M);
 MSE_L_M = mean((signal - xq_L_M).^2);
 results = [results; N, SQNR_uni, SQNR_L_M];
 results_mse = [results_mse; N, MSE_uni, MSE_L_M];
   % Plot original and quantized signals
    subplot(length(N_values), 2, plot_counter);
    plot(signal, 'b-', 'DisplayName', 'Original Signal');
    hold on;
    plot(s_quantized_uni, 'r--', 'DisplayName', ['Uniform Quantized (N = ',
num2str(N), ')']);
    xlabel('Sample Index');
    ylabel('Amplitude');
    legend;
    title(['Uniform Quantization for N = ', num2str(N)]);
    grid on;
    plot_counter = plot_counter + 1;

    subplot(length(N_values), 2, plot_counter);
    plot(signal, 'b-', 'DisplayName', 'Original Signal');
```

```
    hold on;
    plot(xq_L_M, 'g--', 'DisplayName', ['Lloyd-Max Quantized (N = ', num2str(N),
')']);
    xlabel('Sample Index');
    ylabel('Amplitude');
    legend;
    title(['Lloyd-Max Quantization for N = ', num2str(N)]);
    grid on;
    plot_counter = plot_counter + 1;
end

 uni_duration = length(s_quantized_uni) / fs;
 L_M_duration = length(xq_L_M) / fs;
 % Play sound
 fprintf('Now playing for N = %d using Uniform Quantizer...\n', N);
 sound(s_quantized_uni, fs);
 pause(uni_duration);
 fprintf('Now playing for N = %d using Lloyd Max Quantizer...\n', N);
 sound(xq_L_M, fs);
 pause(L_M_duration);

hold off;
%SQNR
figure;
plot(results(:, 1), results(:, 2), '-o', 'LineWidth', 2, 'DisplayName', 'Uniform
Quantizer');
hold on;
plot(results(:, 1), results(:, 3), '-s', 'LineWidth', 2, 'DisplayName', 'Lloyd-Max
Quantizer');
hold off;
set(gca, 'XTick', results(:, 1));
xticks(results(:, 1));
xticklabels({'N = 2', 'N = 4', 'N = 8'});
legend('Location', 'best');
xlabel('# Bits (N)');
ylabel('SQNR (dB)');
title('Comparison of SQNR between Uniform and  Lloyd-Max Quantizers');
grid on;

disp('Results of SQNR comparison:');
disp(table(results(:, 1), results(:, 2), results(:, 3), ...
 'VariableNames', {'N (Bits)', 'SQNR Uniform (dB)', 'SQNR LloydMax (dB)'}));

%MSE

figure;
plot(results_mse(:, 1), results_mse(:, 2), '-o', 'LineWidth', 2, 'DisplayName',
'Uniform Quantizer');
hold on;
plot(results_mse(:, 1), results_mse(:, 3), '-s', 'LineWidth', 2, 'DisplayName',
'Lloyd-Max Quantizer');
hold off;
set(gca, 'XTick', results_mse(:, 1));
xticks(results_mse(:, 1));
xticklabels({'N = 2', 'N = 4', 'N = 8'});
```

```
legend('Location', 'best');
xlabel('# Bits (N)');
ylabel('MSE');
title('Comparison of MSE between Uniform and Lloyd-Max Quantizers');
grid on;

disp('Results of MSE comparison:');
disp(table(results_mse(:, 1), results_mse(:, 2), results_mse(:, 3), ...
 'VariableNames', {'N (Bits)', 'MSE Uniform', 'MSE LloydMax'}));


function sqnr = calculate_sqnr(original_signal, quantized_signal)
 signal_power = mean(original_signal.^2);
 noise_power = mean((original_signal - quantized_signal).^2);
 sqnr = 10 * log10(signal_power / noise_power); %sqnr se db
end
```

Μέρος Α2

1, 2, 3,4 (χωρισμένα σε sections για καλύτερη κατανόηση, μαζί με τις επιμέρους συναρτήσεις)

```
load source.mat;
plot(t)
%sound(t); not clear
%%
load source.mat;
p = 5;
[R, r] = Rx(p, t);
%%
a = R\r;
[a_kvantismenes_perioxes, a_centers] = kvantistis(a, 8, -2, 2);
a_kvantismena = a_centers(a_kvantismenes_perioxes);
%%
N = 4;
min_value = -3.5;
max_value = 3.5;
[kwdikopoiimena, centers, a, y] = kwdikopoiitis_dpcm(t, p, N, min_value, max_value);
anakataskevi = apokwdikopoiitis_dpcm(kwdikopoiimena, a, centers);
%%
t';
plot(anakataskevi);
%%

%2
for p = [5, 10]
    for N = 1:3
        [encoded, centers, a, y] = kwdikopoiitis_dpcm(t, p, N, min_value, max_value);
        figure;
        plot(t);
        hold on
        plot(y);
```

```
        hold off
    end
end
%%
MSE = zeros(10, 3);
for p = 5:10
    for N = 1:3
        [kwdikopoiimena, centers, a, y] = kwdikopoiitis_dpcm(t, p, N, min_value,
max_value);
        anakataskevi = apokwdikopoiitis_dpcm(kwdikopoiimena, a, centers);
        MSE(p, N) = 1/size(anakataskevi, 2) * sum((t - anakataskevi').^2);
    end
end

% Create heatmap for MSE
figure;
imagesc(MSE(5:10, 1:3));
colorbar;
xlabel('N');
ylabel('p');
title('MSE Heatmap');
set(gca, 'YTick', 1:6, 'YTickLabel', 5:10);
set(gca, 'XTick', 1:3, 'XTickLabel', 1:3);
%%

%4
for p = [5, 10]
    for N = 1:3
        [kwdikopoiimena, centers, a, y] = kwdikopoiitis_dpcm(t, p, N, min_value,
max_value);
        reconstructed = apokwdikopoiitis_dpcm(kwdikopoiimena, a, centers);

        figure;
        plot(anakataskevi);
    end
end
%%
%og print
figure;
plot(t);

function [y_sfalma_kvantismeno, centers, a_kvantismena, y_sfalma]  =
kwdikopoiitis_dpcm(x, p, N, min_value, max_value)
%x: sima pros kwdikopoiisi p: parel8ontikes times deigmatos gia provlepsi
%y_sfalma_kvantismeno: to kvantismeno sfalma pou 8a stalei
%a_kvantismena: sintelestes Rx pou evgale o kwdikopoiitis
min_value = -3.5;
max_value = 3.5;

%Ipologizw posotites gia provelsi
[R,r] = Rx(p,x);
a = R\r;
[a_kvantismena, a_centers] = kvantistis(a, 8, -2, 2);
a_kvantismena = a_centers(a_kvantismena);
```

```
% pros8etw stoixeia stin arxi tis mnimis gia na doulepsoun oi prwtes p
% provlepseis
y_memory = zeros(size(x,1)+p,1);
y_sfalma_kvantismeno = zeros(size(x));
y_sfalma = zeros(size(x));

x = [ zeros(p,1); x ];
%kwdikopoiisi
for i  =  p + 1 : size(x,1)
    provlepsi = provleptis(a_kvantismena,y_memory(i-p:i-1));
    y_sfalma(i-p) = x(i) - provlepsi;
    [y_sfalma_kvantismeno(i-p), centers] = kvantistis(y_sfalma(i-p), N, min_value,
max_value);
    y_memory(i) = provlepsi + centers(y_sfalma_kvantismeno(i-p));
end
end
function anakataskevi = apokwdikopoiitis_dpcm(kwdikopoiimena, a, centers)
% kwdikopoiimena : sima pros kwdikopoiisi    a: sintelestes gia provepsi
% pros8etw stoixeia stin arxi tis mnimis gia na doulepsoun oi prwtes p
% provlepseis
p = size(a,1);
y_memory = zeros(size(kwdikopoiimena,1)+p,1);
anakataskevi = size(kwdikopoiimena);
kwdikopoiimena = [zeros(p,1); kwdikopoiimena];

%kwdikopoiisi
for i  =  p + 1 : size(kwdikopoiimena,1)
    provlepsi = provleptis(a,y_memory(i-p:i-1));
    anakataskevi(i-p) = centers(kwdikopoiimena(i)) + provlepsi;
    y_memory(i) = anakataskevi(i-p);
end
%anakataskevi : anakataskevasmeno sima
end
function [kvantismena, centers] = kvantistis(x, N, min_value, max_value)

min_value = -3.5;
max_value = 3.5;

size_vima = (max_value - min_value) / 2^N; % Vriskw mege8os vimatos kvantisis
kvantismena = ones(size(x))*2^N; %Ftiaxnw tin mikroteri timi

%Evresi swstis perioxis
for i =1:size(x,1)
    for j = 1:2^N
        if x(i) >= max_value - size_vima*j
            kvantismena(i) = j;
            break;
        end
    end
end
centers = max_value - size_vima/2 - size_vima * (0:2^N-1)'; % Ypologismos centers
end
function provlepsi = provleptis(a, memory)
provlepsi = sum(a.* flip(memory));
end
```

```
function [R, r]  = Rx(p, x)

R = zeros(p);
r = zeros(p,1);

N = size(x,1);

for i = 1:p
    sum = 0;
    for n = p+1 : N
       sum = sum + x(n) * x(n-i);
    end
    r(i) =  1/(N-p) *sum;
end

for i = 1:p
    for j = 1:p
        sum = 0;
        for n = p+1 : N
           sum = sum + x(n-j) * x(n-i);
        end
        R(i,j) =  1/(N-p) *sum;
    end
end


end
```

Μέρος Β

1,2,3

```
% Initialization
modulation_levels = [2, 8];
bit_length = 100000;
SNR_dB = 0:2:20;
BER = zeros(length(modulation_levels), length(SNR_dB));
SER = zeros(length(modulation_levels), length(SNR_dB));
BER_gray = zeros(1, length(SNR_dB));
plotStyles = {
 {'b-o', 'LineWidth', 1.5, 'MarkerSize', 8}
 {'r-o', 'LineWidth', 1.5, 'MarkerSize', 8}
};

results = [];

for m_idx = 1:length(modulation_levels)
    % Initialize parameters for M-PAM
    M = modulation_levels(m_idx);
    bit_rate = 1e6;
    symbol_rate = 250e3;
    symbol_duration = 1 / symbol_rate;
    sampling_frequency = 10 * symbol_rate;
```

```
    energy_per_symbol = 1;
    energy_per_bit = energy_per_symbol / log2(M);
    carrier_frequency = 2.5e6;
    distance = 2 / sqrt(3 * (M^2 - 1));
    time_vector = 0:1/sampling_frequency:symbol_duration-(1/sampling_frequency);
    bits_per_symbol = log2(M);

    % Length adjustment to make it divisible by the sequence
    N = floor(bit_length / bits_per_symbol);
    adjusted_bit_length = N * bits_per_symbol;

    % Generate random bit sequence
    bit_sequence = rand(1, adjusted_bit_length) > 0.5;
    bit_sequence = double(bit_sequence);

    % Map bits to M-PAM symbols
    symbols = zeros(1, N);
    for k = 1:N
        bit_group = bit_sequence((k-1)*bits_per_symbol+1:k*bits_per_symbol);
        symbols(k) = (bi2de(bit_group, 'left-msb') * 2 - (M - 1)) * distance;
    end

    square_pulse = ones(1, length(time_vector));

    % Modulate the signal with the carrier
    transmitted_signal = zeros(1, N * length(time_vector));
    for k = 1:N
        baseband_signal = symbols(k) * square_pulse;
        transmitted_signal((k-1)*length(time_vector)+1:k*length(time_vector)) =
symbols(k) * cos(2 * pi * carrier_frequency * time_vector);
    end

    % Pass through the noisy channel
    for snr_idx = 1:length(SNR_dB)
        noise_variance = energy_per_symbol / (2 * (10^(SNR_dB(snr_idx)/10)));
        noise = sqrt(noise_variance) * randn(1, length(transmitted_signal));
        received_signal = transmitted_signal + noise;

        % Demodulate the signal
        demodulated_signal = zeros(1, N);
        for k = 1:N
            baseband_signal = received_signal((k-
1)*length(time_vector)+1:k*length(time_vector)) .* cos(2 * pi * carrier_frequency *
time_vector);
            demodulated_signal(k) = trapz(time_vector, baseband_signal .*
square_pulse) / symbol_duration;
        end

        % Map symbols back to bits
        demodulated_bits = zeros(1, N * bits_per_symbol);
        for k = 1:N
            [~, closest_idx] = min(abs(demodulated_signal(k) - ((-(M-1):2:(M-1)) *
distance)));
            demod_symbol = closest_idx - 1;
```

```
            demodulated_bits((k-1)*bits_per_symbol+1:k*bits_per_symbol) =
de2bi(demod_symbol, bits_per_symbol, 'left-msb');
        end

        % Calculate BER
        BER(m_idx, snr_idx) = sum(bit_sequence ~= demodulated_bits) /
length(bit_sequence);

        % Map symbols back to bits for SER calculation
        detected_symbols = zeros(1, N);
        M_levels = (-(M-1):2:(M-1)) * distance; % M-PAM levels
        for k = 1:N
            [~, closest_idx] = min(abs(demodulated_signal(k) - M_levels));
            detected_symbols(k) = M_levels(closest_idx);
        end
        % Calculate SER
        symbol_errors = sum(symbols ~= detected_symbols);
        SER(m_idx, snr_idx) = symbol_errors / N;

        % Store results for displaying later
        results = [results; M, SNR_dB(snr_idx), BER(m_idx, snr_idx), SER(m_idx,
snr_idx), 0];
    end

    if M == 8
        % Gray Coding for M=8
        pam_levels = distance * (-(M-1):2:(M-1));

        % Map bits to Gray-coded M-PAM symbols
        symbols_gray = zeros(1, N);
        for k = 1:N
            bit_group = bit_sequence((k-1)*bits_per_symbol+1:k*bits_per_symbol);
            gray_code = binary_to_gray(bit_group);
            pam_signal = gray_to_pam(gray_code, pam_levels);
            symbols_gray(k) = pam_signal;
        end

        transmitted_signal_gray = zeros(1, N * length(time_vector));
        for k = 1:N
            baseband_signal = symbols_gray(k) * square_pulse;
            transmitted_signal_gray((k-
1)*length(time_vector)+1:k*length(time_vector)) = symbols_gray(k) * cos(2 * pi *
carrier_frequency * time_vector);
        end

        % Pass through the noisy channel
        for snr_idx = 1:length(SNR_dB)
            noise_variance = energy_per_symbol / (2 * (10^(SNR_dB(snr_idx)/10)));
            noise = sqrt(noise_variance) * randn(1, length(transmitted_signal_gray));
            received_signal_gray = transmitted_signal_gray + noise;

            % Demodulate the Gray-coded signal
            demodulated_signal_gray = zeros(1, N);
            for k = 1:N
```

```
                baseband_signal = received_signal_gray((k-
1)*length(time_vector)+1:k*length(time_vector)) .* cos(2 * pi * carrier_frequency *
time_vector);
                demodulated_signal_gray(k) = trapz(time_vector, baseband_signal .*
square_pulse) / symbol_duration;
            end

            demodulated_bits_gray = zeros(1, N * bits_per_symbol);
            for k = 1:N
                received_signal_value = demodulated_signal_gray(k);
                [~, closest_idx] = min(abs(pam_levels - received_signal_value));
                received_pam_symbol = pam_levels(closest_idx);
                gray_code = pam_to_gray(received_pam_symbol, pam_levels,
bits_per_symbol);
                decoded_bits_group = gray_to_binary(gray_code);
                demodulated_bits_gray((k-1)*bits_per_symbol+1:k*bits_per_symbol) =
decoded_bits_group;
            end

            % Calculate BER for Gray Coding
            BER_gray(snr_idx) = sum(bit_sequence ~= demodulated_bits_gray) /
length(bit_sequence);

            % Store Gray-coded BER in results
            results(results(:, 1) == 8 & results(:, 2) == SNR_dB(snr_idx), 5) =
BER_gray(snr_idx);
        end
    end
end

% Display results in a table format
disp('Results of BER and SER comparison:');
disp(table(results(:, 1), results(:, 2), results(:, 3), results(:, 4), results(:, 5),
...
 'VariableNames', {'M', 'SNR(dB)', 'BER', 'SER', 'BER Gray'}));

% Create BER plot
figure('Name', 'M-PAM BER Performance Comparison');
for m_idx = 1:length(modulation_levels)
    if modulation_levels(m_idx) == 8
        semilogy(SNR_dB, BER(m_idx, :), 'r-o', 'LineWidth', 1.5, 'MarkerSize', 8,
'DisplayName', sprintf('M = %d', modulation_levels(m_idx)));
    else
        semilogy(SNR_dB, BER(m_idx, :), plotStyles{m_idx}{:}, 'DisplayName',
sprintf('M = %d', modulation_levels(m_idx)));
    end
    hold on;
end
semilogy(SNR_dB, BER_gray(:), 'g--s', 'LineWidth', 1.5, 'MarkerSize', 8,
'DisplayName', sprintf('8-PAM Gray'));
grid on;
xlabel('SNR(dB)', 'FontSize', 12);
ylabel('Bit Error Rate (BER)', 'FontSize', 12);
title('BER: Standard vs Gray', 'FontSize', 14);
legend('show', 'Location', 'southwest');
```

```
axis([min(SNR_dB) max(SNR_dB) 1e-5 1]);

% Create SER plot
figure('Name', 'M-PAM SER Performance');
lineStyles = {'-o', '-s'};
colors = {'b', 'r'};
for m_idx = 1:length(modulation_levels)
    semilogy(SNR_dB, SER(m_idx, :), [colors{m_idx}, lineStyles{m_idx}], 'LineWidth',
1.5, 'MarkerSize', 8, 'DisplayName', [num2str(modulation_levels(m_idx)), '-PAM']);
    hold on;
end
grid on;
xlabel('SNR (dB)', 'FontSize', 12);
ylabel('Symbol Error Rate (SER)', 'FontSize', 12);
title('SER Performance Comparison of M-PAM Systems', 'FontSize', 14);
legend('show', 'Location', 'southwest');
axis([min(SNR_dB) max(SNR_dB) 1e-5 1]);
hold off;

% Helper functions
function pam_signal = gray_to_pam(gray_code, pam_levels)
    decimal_index = bi2de(gray_code, 'left-msb'); % Find the PAM level index
    pam_signal = pam_levels(decimal_index + 1); % Get the PAM level
end

function gray_code = binary_to_gray(binary_num)
    gray_code = zeros(size(binary_num));
    % Copy the MSB
    gray_code(1) = binary_num(1);
    % XOR the current and previous bit
    for i = 2:length(binary_num)
        gray_code(i) = xor(binary_num(i-1), binary_num(i));
    end
end

function gray_code = pam_to_gray(pam_signal, pam_levels, bits_per_symbol)
    % Find the index of the PAM level
    idx = find(pam_levels == pam_signal, 1) - 1;
    % Convert to binary representation
    gray_code = de2bi(idx, bits_per_symbol, 'left-msb');
end

function binary_num = gray_to_binary(gray_code)
    binary_num = zeros(size(gray_code));
    % Copy the MSB
    binary_num(1) = gray_code(1);
    % XOR the current and previous bit
    for i = 2:length(gray_code)
        binary_num(i) = xor(binary_num(i-1), gray_code(i));
    end
end
```