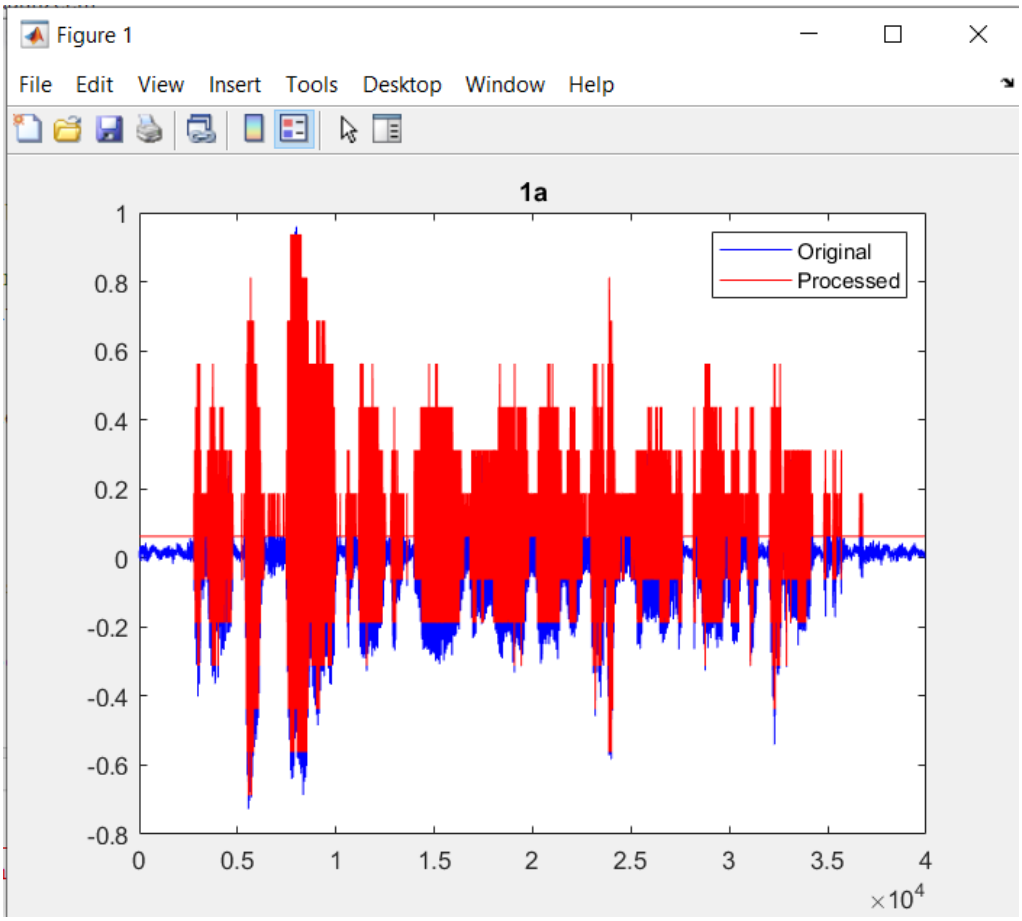
1η Εργασία Ψηφιακές Τηλεπικοινωνίες

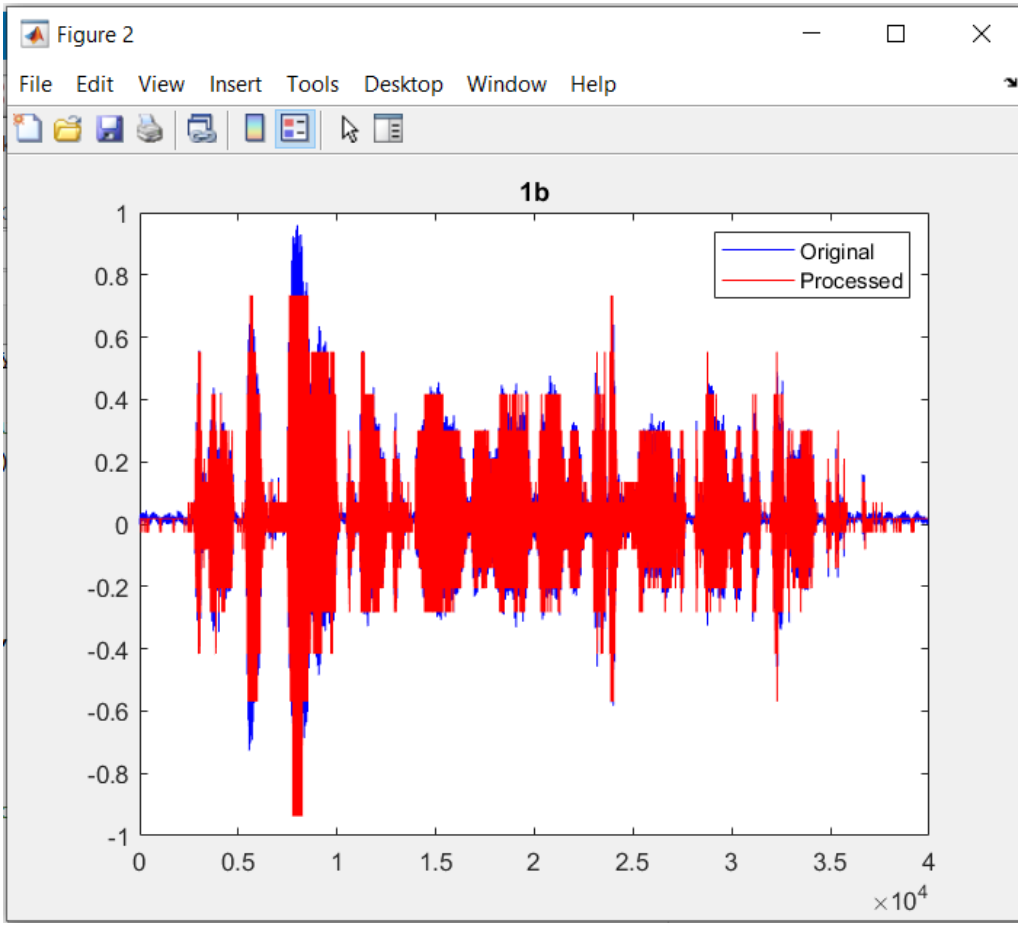
Μέρος Α

Α1

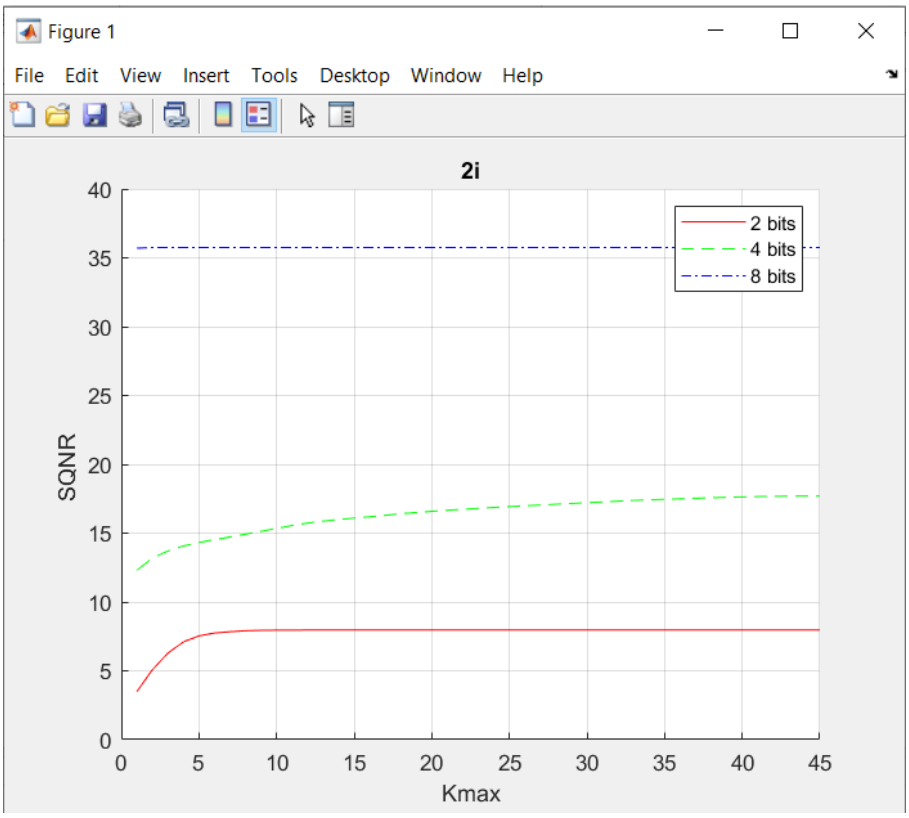
1a) Σχήμα PCM χρησιμοποιώντας τον ομοιόμορφο κβαντιστή.



1b) Σχήμα PCM χρησιμοποιώντας τον μη ομοιόμορφο με χρήση του αλγόριθμου Lloyd-Max.



2i) Σχήμα κβάντισης που δείχνει πώς μεταβάλλεται το SQNR σε σχέση με το Kmax.



Παρατηρούμε πως όσο αυξάνεται το N το SQNR είναι υψηλότερο(άρα και υψηλότερο Kmax). Αυτό συμβαίνει διότι όσα περισσότερα bits έχουμε θα υπάρχει μεγαλύτερη ακρίβεια.

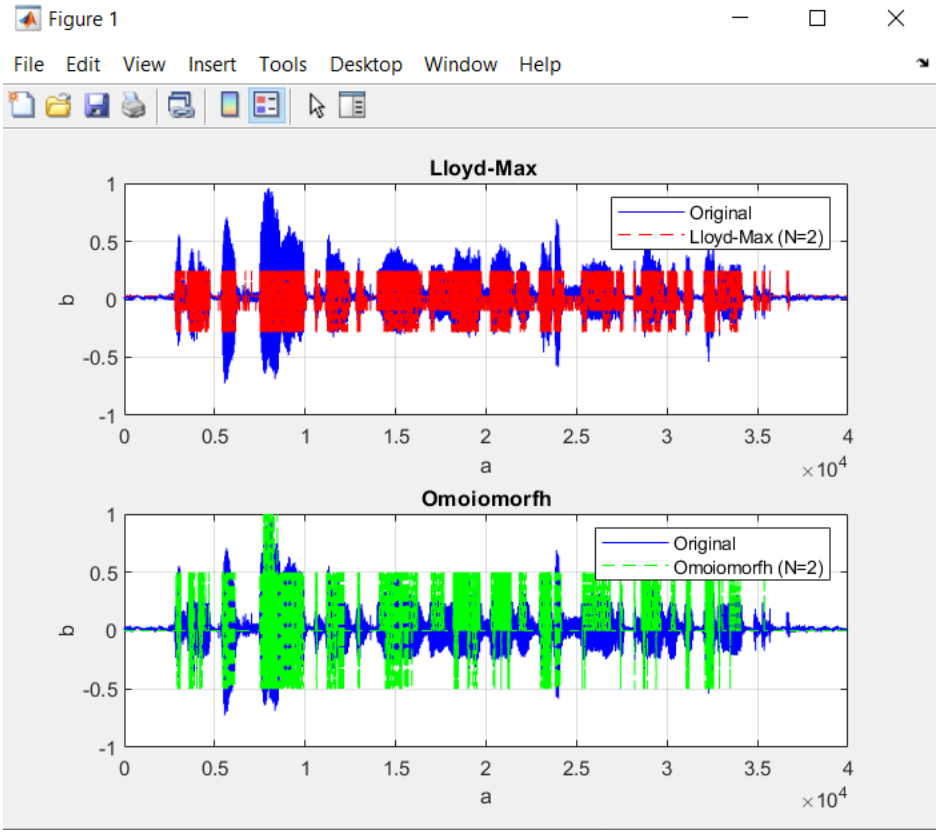
2ii) Σύγκρίση τιμής του SQNR μετά από Kmax επαναλήψεις.

3×3 table

Bits	SQNR_Uniform	SQNR_Lloyd_Max
2	2.6818	7.9644
4	12.547	17.7
8	35.57	35.749

Από τον πίνακα παρατηρούμε ότι όσο αυξάνονται τα bits η διαφορά του ομοιόμορφου και του μη ομοιόμορφου κβαντιστή μειώνεται.

2iii)Σύγκριση σημάτων.



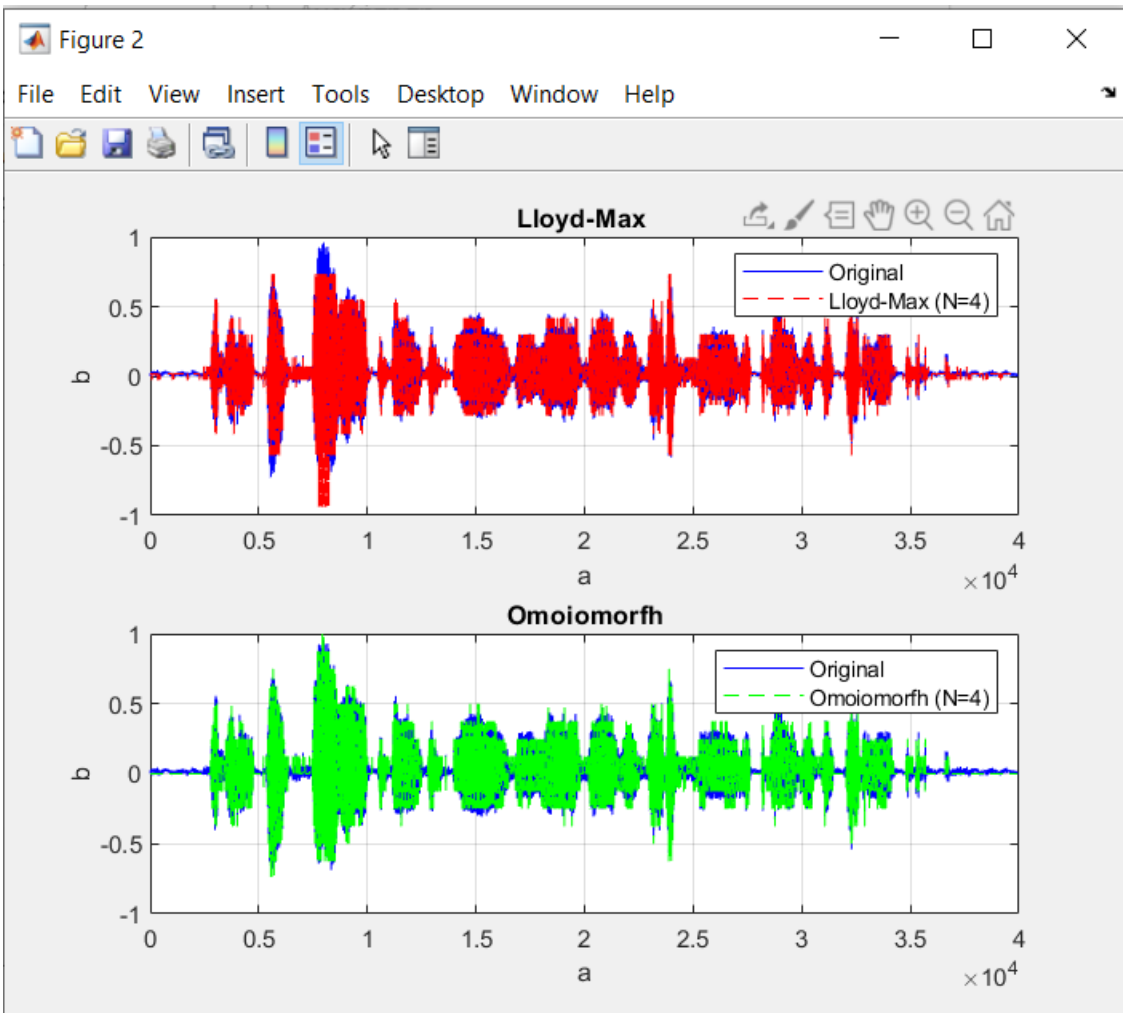
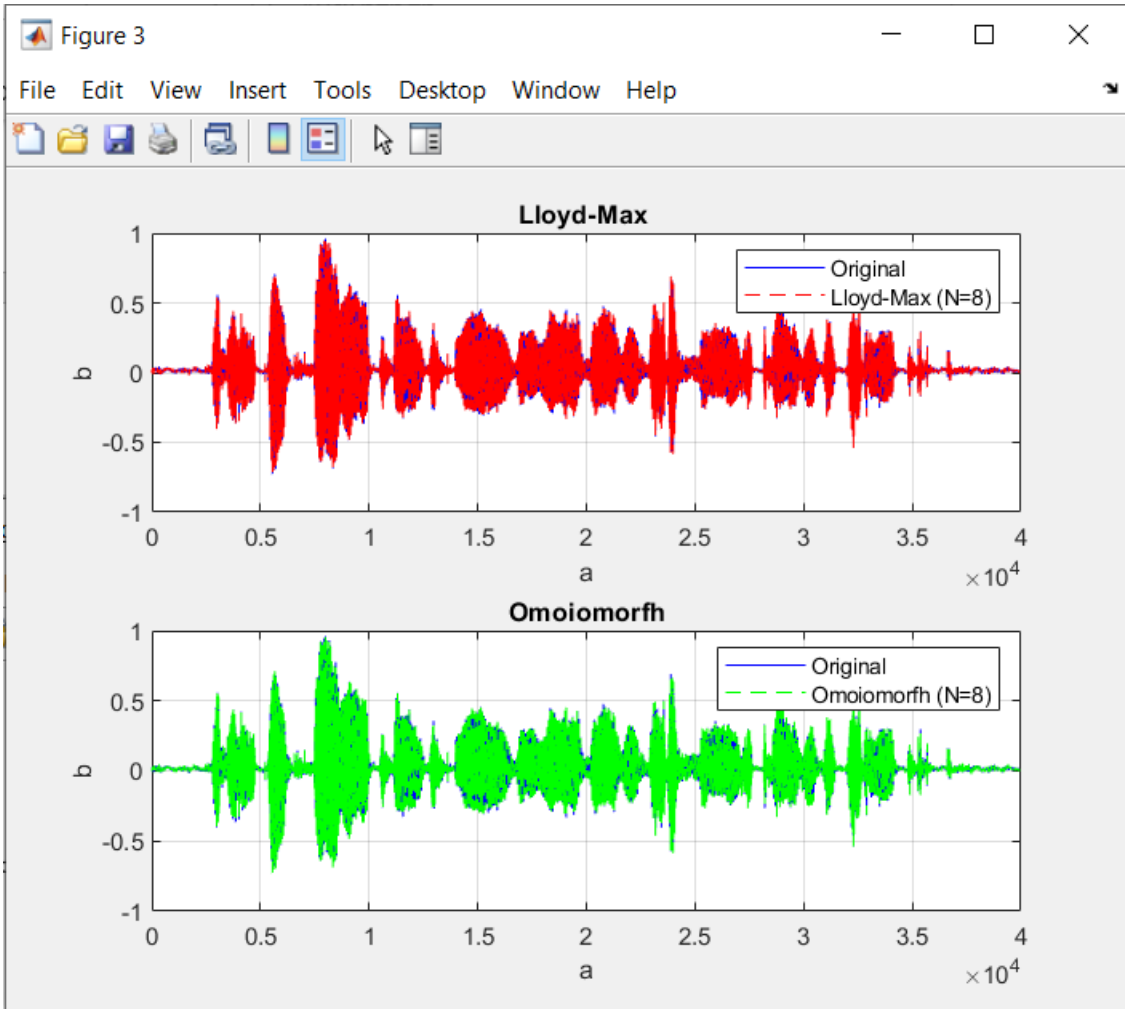
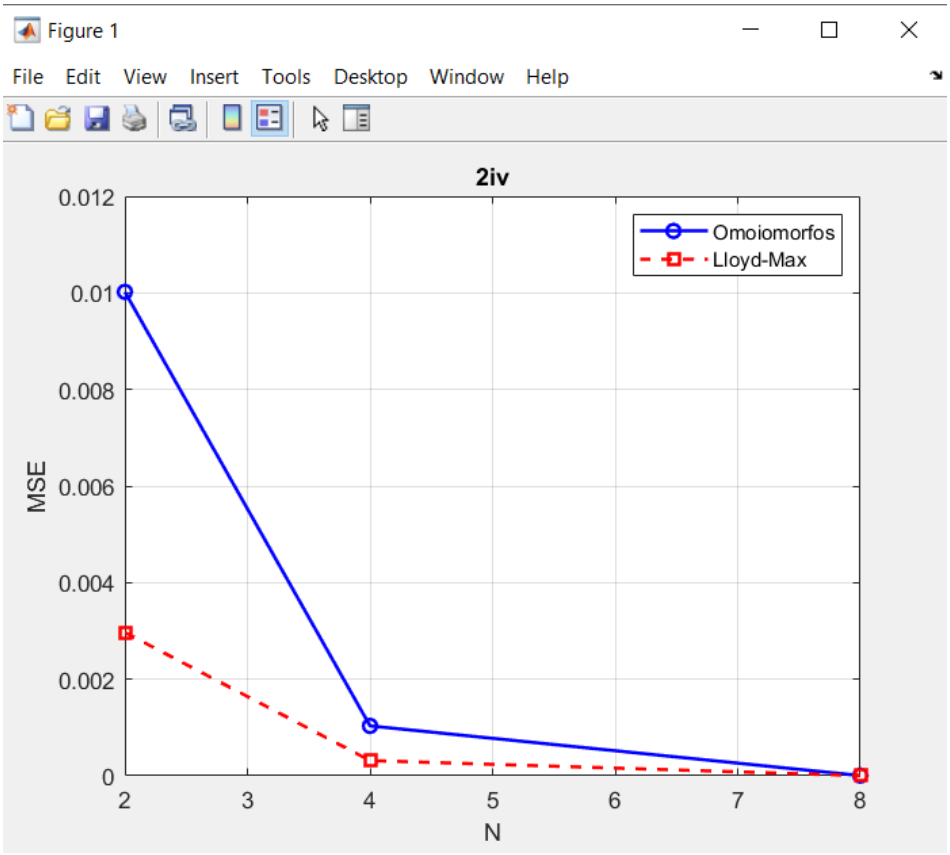


Figure 2
File Edit View Insert Tools Desktop Window Help
Lloyd-Max
Original
Lloyd-Max (N=4)
b
a
×10^4
Omoiomorfh
Original
Omoiomorfh (N=4)
b
a
×10^4



Παρατηρούμε πως και η ποιότητα του ήχου αλλά και οι κυματομορφές πλησιάζουν το αρχικό σήμα όσο αυξάνεται το Ν.

2iv)MSE για τους 2 κβαντιστές.



Παρατηρούμε ότι όσο αυξάνεται το N, το MSE μειώνεται. Βλέπουμε ότι για N=2 οι κβαντιστές έχουν μεγάλη διαφορά ενώ για N=8 οι κβαντιστές συγκλίνουν.

Μέρος Β

1.Παρακάτω υλοποιείται το σύστημα M-PAM:

-> Διαμόρφωση M-PAM

```
function S_m = diamorfosi_PAM(symbols, M)
   %Parameters
   Tsymbol = 40;
   Tsample = 1;
   Tc = 4;
   Es = 1;
   fc = 1 / Tc;
   Rs=250;

   % Calculating A
   A = 0;
   for m = 1:M
      A = A + (2 * m - (M + 1))^2;
   end
   A = 1 / sqrt(A / M);

   % Διαμόρφωση PAM
   t = 1:Tsymbol;
   S_m = zeros(length(symbols), Tsymbol);
   for i = 1:length(symbols)
      Am = (2 * (symbols(i) + 1) - (M + 1)) * A;
```

```
        S_m(i, :) = Am * sqrt(2 * Es / Tsymbol) * cos(2 * pi * fc * t);
    end
end
```

->Κανάλι AWGN

```
function sn = noise(S_m, SNR, M)
    % AWGN parameters
    Eb = Es / M;
    Es = 1;
    N0 = Eb / (10^(SNR / 10));

    noise = sqrt(N0 / 2) * randn(size(S_m));  %Για την παραγωγή του θορύβου
    sn = S_m + noise;
end
```

->Αποδιαμορφωτής M-PAM

```
function r = apodiamorphwtis_PAM(sn)
    % Parameters
    Tsymbol = 40;
    Tsample = 1;
    Tc = 4;
    fc = 1 / Tc;
    t = 1:Tsymbol;
```

```
    % Φέρουσα
    c = sqrt(2 / Tsymbol) * cos(2 * pi * fc * t)';
    r = sn * c;
end
```

->Φωρατής M-PAM

```
function new = foratis_PAM(r, M)
    % Υπολογισμός του A
    A = 0;
    for i = 1:M
        A = A + (2 * i - (M + 1))^2;
    end
    A = 1 / sqrt(A / M);

    % Ανίχνευση καινούργιων συμβόλων
    new = zeros(size(r));
    for i = 1:length(r)
        min_distance = inf;
        for m = 1:M
            Am = A * (2 * m - (M + 1));
            distance = (r(i) - Am)^2;
            if distance < min_distance
                min_distance = distance;
                new(i) = m - 1;
```

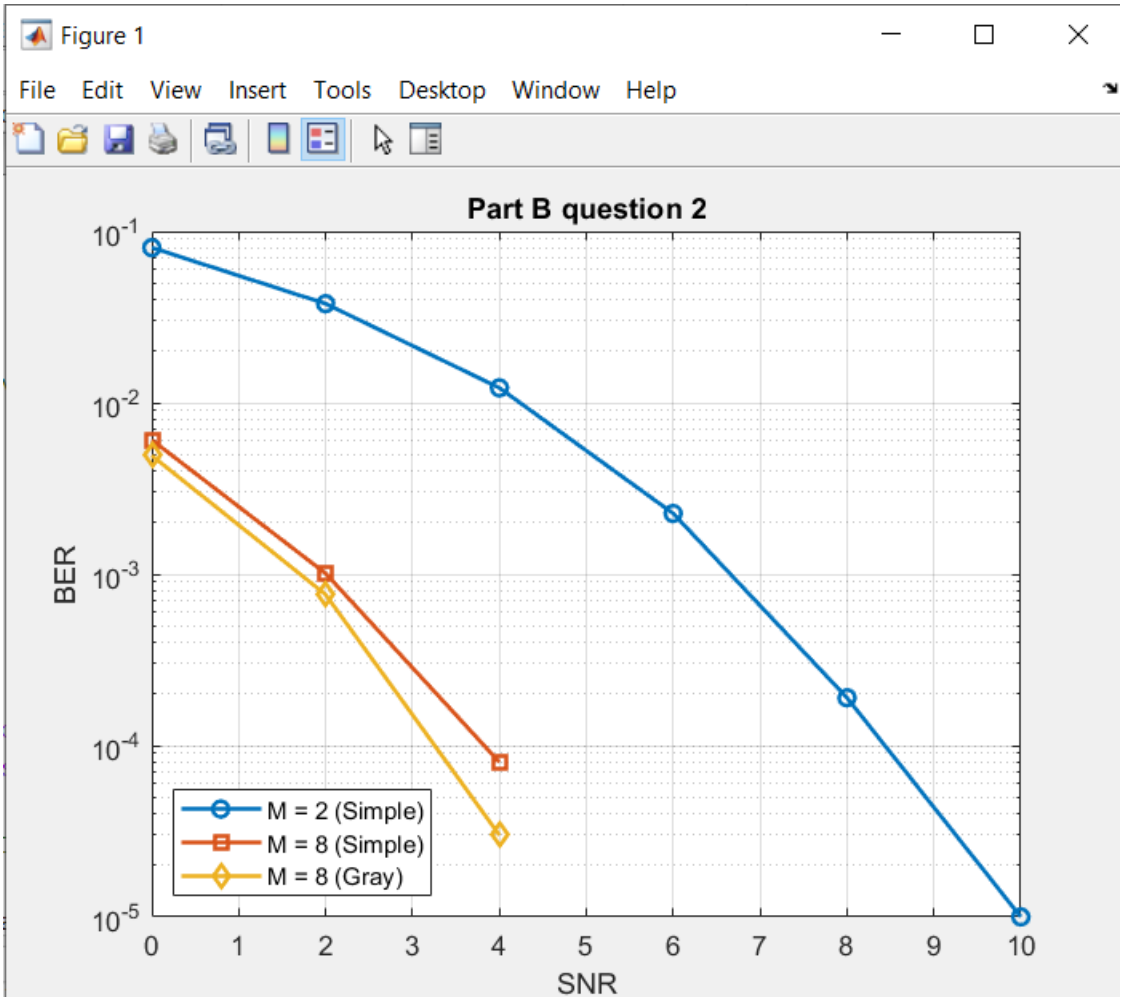
```
        end
      end
   end
end
```

->Μετρήσεις BER-SER

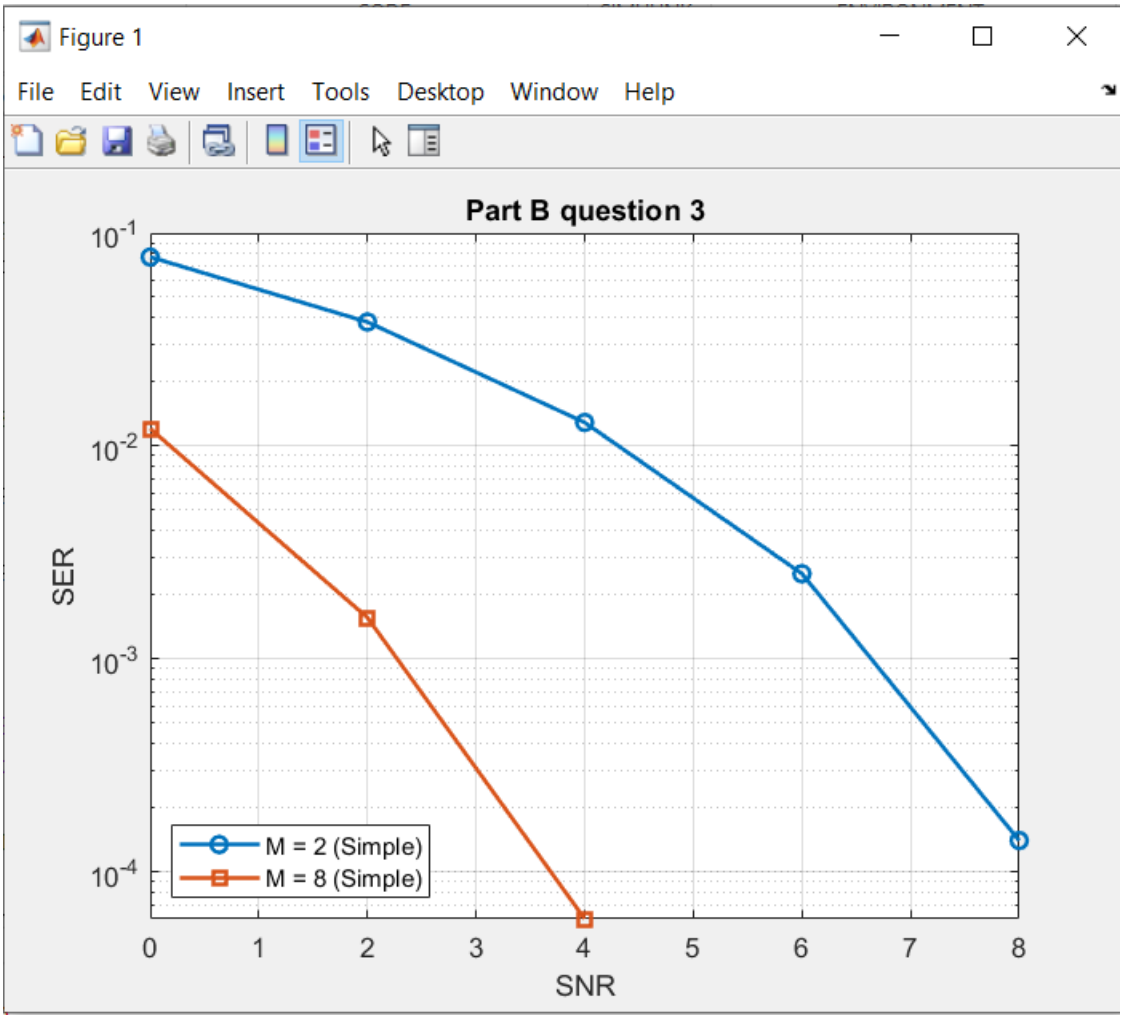
```
%SER
function ser1 = ser(symbols, new)
   errors = symbols ~= new;
   ser1 = sum(errors) / length(symbols);
end

%BER
function ber1 = ber(bits, bits1)
   errors = bits ~= bits1;
   ber1 = sum(errors) / length(bits);
end
```

2) Στο παρακάτω γράφημα απεικονίζεται η καμπύλη BER για M = 2 & 8 για απλή κωδικοποίηση και για M = 8 αν τα σύμβολα στην αντιστοίχιση κωδικοποιούνταν κατά Gray.



3) Στο παρακάτω γράφημα υλοποιείται SER για M = 2,8 για απλή κωδικοποίης.



Κώδικες

A1)

α)

```
function [xq, centers] = my_quantizer (x, N, min_value, max_value)
   x(x > max_value) = max_value;  % Αποδεκτές τιμές
   x(x < min_value) = min_value; % Αποδεκτές τιμές

   Delta = (max_value - min_value) / (2^N); % Βήματα κβάντισης

   centers = min_value + Delta/2 : Delta : max_value - Delta/2; % Κέντρα Κβάντισης

   % Περιοχή στην οποία ανήκει κάθε δείγμα
   xq = round((x - min_value) / Delta) + 1;
   xq(xq < 1) = 1;
   xq(xq > 2^N) = 2^N;
end

% Φόρτωση του ηχητικού αρχείου
[y, fs] = audioread('speech.wav');  % Φόρτωση σήματος στην Matlab

min_value = -1;
max_value = 1;
N = 4;
```

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

quantized_signal = centers(xq); % Δημιουργία του σήματος
sound(quantized_signal, fs);

% Εμφάνιση γραφημάτων
figure;
plot(y, 'b'); hold on; % Αρχικό σήμα
plot(quantized_signal, 'r'); % Κβαντισμένο σήμα
legend('Original', 'Processed');
title('1a');

1b)
function [xq, centers, D] = Lloyd_Max(x, N, min_value, max_value, epsilon)
   x(x > max_value) = max_value;  % Αποδεκτές τιμές
   x(x < min_value) = min_value;  % Αποδεκτές τιμές

   Delta = (max_value - min_value) / (2^N);
   centers = min_value + Delta/2 : Delta : max_value - Delta/2;

   % Επανάληψη του Lloyd-Max
   D_prev = inf;
   D = [];
   while true
```

```
    boundaries = [(min_value + centers(1))/2, (centers(1:end-1) +
centers(2:end))/2, (max_value + centers(end))/2]; % Όρια περιοχών

    % Κβάντιση
    xq = zeros(size(x));
    distortion_sum = 0; % Άθροισμα σφαλμάτων
    count = 0;
    for i = 1:length(centers)
        idx = (x >= boundaries(i) & x < boundaries(i+1));
        xq(idx) = i;

        % Υπολογισμός νέου κέντρου
        samples = x(idx);
        if ~isempty(samples)
            centers(i) = mean(samples);
        end

        % Υπολογισμός παραμόρφωσης
        distortion_sum = distortion_sum + sum((samples - centers(i)).^2);
        count = count + numel(samples);
    end
    xq(xq == 0) = 1;

    distortion = distortion_sum / count; % Υπολογίζουμε την παραμόρφωση
```

```
        D = [D, distortion];

        % Έλεγχος σύγκλισης
        if abs(D_prev - distortion) < epsilon
            break;
        end
        D_prev = distortion;
    end
end



% Φόρτωση του ηχητικού αρχείου
[y, fs] = audioread('speech.wav');

N = 4;
min_value = -1;
max_value = 1;
epsilon = 1e-6;



[xq, centers, D] = Lloyd_Max(y, N, min_value, max_value, epsilon); % Κλήση της συνάρτησης Lloyd_Max

quantized_signal = centers(xq); % Κβαντισμένο σήμα
sound(quantized_signal, fs);
```

```
% Σχεδίαση παραμόρφωσης
figure;
plot(1:length(D), 10*log10(D), '-o');
xlabel('Iterations');
ylabel('SQNR (dB)');
title('Lloyd-Max');

% Σύγκριση αρχικού και κβαντισμένου σήματος
figure;
plot(y, 'b'); hold on;
plot(quantized_signal, 'r');
legend('Original ', 'Processed');
title('1b');

2i) % Φόρτωση του ηχητικού αρχείου
[y, fs] = audioread('speech.wav');

N_values = [2, 4, 8];
min_value = -1;
max_value = 1;
epsilon = 1e-6;

signal_power = mean(y.^2); % Ισχύη σήματος
```

```
figure; hold on;
colors = ['r', 'g', 'b'];
line_styles = {'-', '--', '-.'};

% Υλοποίηση για κάθε N
for idx = 1:length(N_values)
    N = N_values(idx);

    [xq, centers, D] = Lloyd_Max(y, N, min_value, max_value, epsilon);



    if length(D) < 10
        disp(['Warning: Low iterations for N = ', num2str(N)]);
    end

    SQNR = 10 * log10(signal_power ./ D); %Υπολογισμός SQNR

    % Επέκταση του μήκους SQNR για να εμφανίζεται σωστά
    Kmax = 1:length(SQNR);
    if length(Kmax) < 45
        Kmax = [Kmax, (length(Kmax)+1):45];
        SQNR = [SQNR, SQNR(end) * ones(1, 45 - length(SQNR))];
    end
```

```
    % Σχεδίαση καμπύλης
    plot(Kmax, SQNR, 'Color', colors(idx), 'LineStyle', line_styles{idx}, ...
        'DisplayName', sprintf('%d bits', N));
end

% Γράφημα
xlabel('Kmax');
ylabel('SQNR ');
title('2i');
legend('show');
grid on;
hold off;
```

2ii)

```
[y, fs] = audioread('speech.wav'); % Φόρτωση σήματος στην Matlab

N_values = [2, 4, 8];
min_value = -1;
max_value = 1;
epsilon = 1e-6;


signal_power = mean(y.^2); % Ισχύη σήματος
```

```
% Αποτελέσματα για σύγκριση
SQNR_uniform = zeros(size(N_values)); % Ομοιόμορφη κβάντιση
SQNR_lloyd_max = zeros(size(N_values)); % Lloyd-Max

% Υπολογισμός για κάθε N
for idx = 1:length(N_values)
    N = N_values(idx);

    %  Lloyd-Max
    [xq_lloyd, centers_lloyd, D_lloyd] = Lloyd_Max(y, N, min_value, max_value, epsilon);
    SQNR_lloyd_max(idx) = 10 * log10(signal_power / D_lloyd(end)); % Τελευταία παραμόρφωση

    % Ομοιόμορφη κβάντιση
    delta = (max_value - min_value) / (2^N); % Μέγεθος βήματος
    centers_uniform = min_value + delta/2 : delta : max_value - delta/2; % Κέντρα κβάντισης
    % Εύρεση του κοντινότερου επιπέδου κβάντισης
    quantized_signal_uniform = min_value + delta * round((y - min_value) / delta);
    quantized_signal_uniform = max(min(quantized_signal_uniform, max_value), min_value); % Περιορισμός εντός ορίων
    D_uniform = mean((y - quantized_signal_uniform).^2); % Υπολογισμός παραμόρφωσης
    SQNR_uniform(idx) = 10 * log10(signal_power / D_uniform);
```

```
end

% Εμφάνιση αποτελεσμάτων
disp('Αποτελέσματα SQNR (σε dB):');
table(N_values', SQNR_uniform', SQNR_lloyd_max', 'VariableNames', ...
    {'Bits', 'SQNR_Uniform', 'SQNR_Lloyd_Max'})

% Γράφημα για σύγκριση
figure;
bar(N_values, [SQNR_uniform', SQNR_lloyd_max'], 'grouped');
xlabel('Bits ');
ylabel('SQNR ');
title('2ii');
legend('Uniform Quantization', 'Lloyd-Max Quantization');
grid on;
```

2iii)

```
[y, fs] = audioread('speech.wav');

N_values = [2, 4, 8];
min_value = -1;
max_value = 1;
epsilon = 1e-6;
```

```
disp('Αναπαραγωγή αρχικού σήματος...');
sound(y, fs);
pause(length(y) / fs + 2);

for idx = 1:length(N_values)
    N = N_values(idx);

    % Lloyd-Max Κβάντιση
    [xq_lloyd, centers_lloyd, ~] = Lloyd_Max(y, N, min_value, max_value, epsilon);
    quantized_signal_lloyd = centers_lloyd(xq_lloyd);

    % Ομοιόμορφη Κβάντιση
    delta = (max_value - min_value) / (2^N); % Μέγεθος βήματος
    quantized_signal_uniform = min_value + delta * round((y - min_value) / delta);
    quantized_signal_uniform = max(min(quantized_signal_uniform, max_value),
min_value); % Περιορισμός εντός ορίων

    % Αναπαραγωγή κωδικοποιημένων σημάτων
    disp(['Lloyd-Max gia N = ', num2str(N), ' bits...']);
    sound(quantized_signal_lloyd, fs);
    pause(length(y) / fs + 2);

    disp(['Omoiomorfh gia  N = ', num2str(N), ' bits...']);
    sound(quantized_signal_uniform, fs);
    pause(length(y) / fs + 2);
```

```
    % Σχεδίαση κυματομορφών
    figure;
    subplot(2, 1, 1);
    plot(y, 'b-', 'DisplayName', 'Original');
    hold on;
    plot(quantized_signal_lloyd, 'r--', 'DisplayName', ['Lloyd-Max (N=', num2str(N),
')']);
    title('Lloyd-Max ');
    xlabel('a');
    ylabel('b');
    legend show;
    grid on;

    subplot(2, 1, 2);
    plot(y, 'b-', 'DisplayName', 'Original');
    hold on;
    plot(quantized_signal_uniform, 'g--', 'DisplayName', ['Omoiomorfh (N=',
num2str(N), ')']);
    title('Omoiomorfh');
    xlabel('a');
    ylabel('b');
    legend show;
    grid on;
end
```

2iv)

```
[y, fs] = audioread('speech.wav'); % Φόρτωση σήματος στην Matlab


min_value = -1;

max_value = 1;

y = max(min(y, max_value), min_value);


% Ορισμός παραμέτρων

N_values = [2, 4, 8]; % Bits κβάντισης

MSE_uniform = zeros(size(N_values)); % MSE (ομοιόμορφο κβαντιστή)

MSE_lloyd = zeros(size(N_values)); % MSE (Lloyd-Max)


for idx = 1:length(N_values)

    N = N_values(idx);


    % Ομοιόμορφος Κβαντιστής

    delta = (max_value - min_value) / (2^N); % Βήμα κβάντισης

    quantized_uniform = delta * round((y - min_value) / delta) + min_value; % Κβαντισμένο σήμα

    quantized_uniform = max(min(quantized_uniform, max_value), min_value); % Περιορισμός εντός ορίων

    MSE_uniform(idx) = mean((y - quantized_uniform).^2); % Υπολογισμός MSE
```

```
    % Lloyd-Max Κβαντιστής
    [~, centers, D] = Lloyd_Max(y, N, min_value, max_value, 1e-6); % Υπολογισμός Lloyd-Max
    MSE_lloyd(idx) = D(end);
end

% Σχεδίαση καμπυλών MSE
figure;
plot(N_values, MSE_uniform, 'b-o', 'LineWidth', 1.5, 'DisplayName', 'Omoiomorfos');
hold on;
plot(N_values, MSE_lloyd, 'r--s', 'LineWidth', 1.5, 'DisplayName', 'Lloyd-Max ');
title('2iv');
xlabel('N');
ylabel('MSE');
legend('show');
grid on;
```

Β μέρος

```
2)
SNR_values = 0:2:20;     % Τιμές SNR (dB)
M_values = [2, 8];
num_bits = 1e5;
BER_results = zeros(length(M_values), length(SNR_values)); % BER για κάθε M και SNR
BER_gray_results = zeros(1, length(SNR_values));     % BER με Gray για M=8

% Υπολογισμός BER για κάθε M
```

```
for m_idx = 1:length(M_values)
   M = M_values(m_idx);
   for snr_idx = 1:length(SNR_values)
      SNR_dB = SNR_values(snr_idx);

      % 1. Δημιουργία bits
      num_bits_per_symbol = log2(M); % Bits ανά σύμβολο
      num_symbols = floor(num_bits / num_bits_per_symbol); % Αριθμός
συμβόλων
      bits = randi([0, 1], 1, num_symbols * num_bits_per_symbol); %
Προσαρμοσμένα bits

      % 2. Mapper για PAM
      symbols = Mapper(bits, M, 0); % Απλή κωδικοποίηση

      % 3. Διαμόρφωση PAM
      transmitted_signal = modulate_PAM(symbols, M);

      % 4. Προσθήκη Θορύβου
      noisy_signal = add_noise(transmitted_signal, SNR_dB, M);

      % 5. Αποδιαμόρφωση και Ανίχνευση Συμβόλων
      demodulated_signal = demodulate_PAM(noisy_signal);
      detected_symbols = detect_PAM(demodulated_signal, M);

      % 6. Demapper για bits
      detected_bits = Demapper(detected_symbols, M, 0);

      % 7. Υπολογισμός BER
      BER_results(m_idx, snr_idx) = compute_BER(bits, detected_bits);
   end
end

% Υπολογισμός BER για Gray κωδικοποίηση (M=8)
M = 8;
for snr_idx = 1:length(SNR_values)
   SNR_dB = SNR_values(snr_idx);
```

```
    % 1. Δημιουργία bits
    num_bits_per_symbol = log2(M);
    num_symbols = floor(num_bits / num_bits_per_symbol);
    bits = randi([0, 1], 1, num_symbols * num_bits_per_symbol);

    % 2. Mapper για PAM με Gray κωδικοποίηση
    symbols = Mapper(bits, M, 1); % Gray κωδικοποίηση

    % 3. Διαμόρφωση PAM
    transmitted_signal = modulate_PAM(symbols, M);

    % 4. Προσθήκη Θορύβου
    noisy_signal = add_noise(transmitted_signal, SNR_dB, M);

    % 5. Αποδιαμόρφωση και Ανίχνευση Συμβόλων
    demodulated_signal = demodulate_PAM(noisy_signal);
    detected_symbols = detect_PAM(demodulated_signal, M);

    % 6. Demapper για bits
    detected_bits = Demapper(detected_symbols, M, 1);

    % 7. Υπολογισμός BER
    BER_gray_results(snr_idx) = compute_BER(bits, detected_bits);
end

% Σχεδίαση Γραφημάτων
figure;
semilogy(SNR_values, BER_results(1, :), '-o', 'LineWidth', 1.5, 'DisplayName', 'M =
2 (Simple)');
hold on;
semilogy(SNR_values, BER_results(2, :), '-s', 'LineWidth', 1.5, 'DisplayName', 'M =
8 (Simple)');
semilogy(SNR_values, BER_gray_results, '-d', 'LineWidth', 1.5, 'DisplayName', 'M =
8 (Gray)');
grid on;
xlabel('SNR (dB)');
```

```
ylabel('Bit Error Rate (BER)');
title('Part B question 2');
legend('Location', 'southwest');


function symbols = Mapper(bits, M, gray)
   % Mapper: Μετατροπή bits σε σύμβολα PAM
   num_bits_per_symbol = log2(M);
   num_symbols = length(bits) / num_bits_per_symbol;
   bits_reshape = reshape(bits, num_bits_per_symbol, num_symbols)';

   symbols = bin2dec(num2str(bits_reshape)); % Μετατροπή binary σε decimal

   % Εφαρμογή Gray κωδικοποίησης αν χρειάζεται
   if gray == 1
      symbols = bin2gray(symbols, 'pam', M); % Gray mapping
   end
end

function bits = Demapper(symbols, M, gray)
   % Demapper: Μετατροπή συμβόλων PAM σε bits
   if gray == 1
      symbols = gray2bin(symbols, 'pam', M);
   end
   num_bits_per_symbol = log2(M);
   bits_binary = dec2bin(symbols, num_bits_per_symbol);
   bits = reshape(bits_binary', 1, numel(bits_binary)) - '0';
end

function transmitted_signal = modulate_PAM(input_symbols, M)
   % Διαμόρφωση PAM
   max_amplitude = M - 1;
   transmitted_signal = 2 * input_symbols - max_amplitude;
end


function noisy_signal = add_noise(signal, SNR_dB, M)
```

```
    % Προσθήκη Gaussian θορύβου (AWGN)
    Eb = 1 / log2(M);  % Ενέργεια ανά bit
    noise_power_density = Eb / (10^(SNR_dB / 10));  % Φασματική πυκνότητα
    noise = sqrt(noise_power_density / 2) * randn(size(signal));  % Gaussian
θόρυβος
    noisy_signal = signal + noise; % Θορυβώδες σήμα
end

function demodulated_signal = demodulate_PAM(noisy_signal)

    demodulated_signal = noisy_signal; % Επιστροφή του ληφθέντος σήματος
end

function detected_symbols = detect_PAM(demodulated_signal, M)
    % Ανίχνευση συμβόλων για PAM
    max_amplitude = M - 1; % Μέγιστο πλάτος
    levels = -max_amplitude:2:max_amplitude; % Επίπεδα πλάτους
    detected_symbols = zeros(size(demodulated_signal));

    for i = 1:length(demodulated_signal)
        [~, idx] = min(abs(levels - demodulated_signal(i))); % Βρίσκουμε το
κοντινότερο επίπεδο
        detected_symbols(i) = idx - 1; % Επιστροφή ως σύμβολο
    end
end

function ber_value = compute_BER(original_bits, detected_bits)
    % Υπολογισμός Bit Error Rate (BER)
    ber_value = sum(original_bits ~= detected_bits) / length(original_bits); % BER
end


3) %

SNR_values = 0:2:20;

M_values = [2, 8];

num_bits = 1e5;
```

```
SER_results = zeros(length(M_values), length(SNR_values)); % SER για κάθε M και SNR

% Υπολογισμός SER για κάθε M
for m_idx = 1:length(M_values)
    M = M_values(m_idx);
    bits_per_symbol = log2(M);
    for snr_idx = 1:length(SNR_values)
        SNR_dB = SNR_values(snr_idx);

        % 1. Δημιουργία bits
        num_symbols = floor(num_bits / bits_per_symbol); % Αριθμός συμβόλων
        bits = randi([0, 1], 1, num_symbols * bits_per_symbol); % Δημιουργία bits

        % 2. Mapper για PAM
        symbols = Mapper(bits, M, 0); % Απλή κωδικοποίηση

        % 3. Διαμόρφωση PAM
        transmitted_signal = modulate_PAM(symbols, M);

        % 4. Προσθήκη Θορύβου
        noisy_signal = add_noise(transmitted_signal, SNR_dB, M);

        % 5. Αποδιαμόρφωση και Ανίχνευση Συμβόλων
        demodulated_signal = demodulate_PAM(noisy_signal);
```

```
        detected_symbols = detect_PAM(demodulated_signal, M);

        % 6. Υπολογισμός SER
        SER_results(m_idx, snr_idx) = compute_SER(symbols, detected_symbols);
    end
end

% Σχεδίαση Γραφημάτων SER
figure;
semilogy(SNR_values, SER_results(1, :), '-o', 'LineWidth', 1.5, 'DisplayName', 'M = 2 (Simple)');
hold on;
semilogy(SNR_values, SER_results(2, :), '-s', 'LineWidth', 1.5, 'DisplayName', 'M = 8 (Simple)');
grid on;
xlabel('SNR (dB)');
ylabel('Symbol Error Rate (SER)');
title('Part B question 3');
legend('Location', 'southwest');


function symbols = Mapper(bits, M, gray)
    % Mapper: Μετατροπή bits σε σύμβολα PAM
    num_bits_per_symbol = log2(M); % Bits ανά σύμβολο
    num_symbols = length(bits) / num_bits_per_symbol; % Αριθμός συμβόλων
```

```
    bits_reshape = reshape(bits, num_bits_per_symbol, num_symbols)'; % Ομαδοποίηση bits σε log2(M)
    symbols = bin2dec(num2str(bits_reshape)); % Μετατροπή binary σε decimal

    % Εφαρμογή Gray κωδικοποίησης αν χρειάζεται
    if gray == 1
        symbols = bin2gray(symbols, 'pam', M); % Gray mapping
    end
end

function transmitted_signal = modulate_PAM(input_symbols, M)
    % Διαμόρφωση PAM
    max_amplitude = M - 1; % Μέγιστο πλάτος
    transmitted_signal = 2 * input_symbols - max_amplitude; % Μετατροπή σε PAM επίπεδα
end

function noisy_signal = add_noise(signal, SNR_dB, M)
    % Προσθήκη Gaussian θορύβου (AWGN)
    Eb = 1 / log2(M);  % Ενέργεια ανά bit
    noise_power_density = Eb / (10^(SNR_dB / 10));  % Φασματική πυκνότητα
    noise = sqrt(noise_power_density / 2) * randn(size(signal));  % Gaussian θόρυβος
    noisy_signal = signal + noise; % Θορυβώδες σήμα
end
```

```
function demodulated_signal = demodulate_PAM(noisy_signal)

    demodulated_signal = noisy_signal; % Επιστροφή του ληφθέντος σήματος
end

function detected_symbols = detect_PAM(demodulated_signal, M)
    % Ανίχνευση συμβόλων για PAM
    max_amplitude = M - 1; % Μέγιστο πλάτος
    levels = -max_amplitude:2:max_amplitude; % Επίπεδα πλάτους
    detected_symbols = zeros(size(demodulated_signal));

    for i = 1:length(demodulated_signal)
        [~, idx] = min(abs(levels - demodulated_signal(i))); % Βρίσκουμε το κοντινότερο επίπεδο
        detected_symbols(i) = idx - 1; % Επιστροφή ως σύμβολο
    end
end

function ser_value = compute_SER(original_symbols, detected_symbols)
    % Υπολογισμός Symbol Error Rate (SER)
    ser_value = sum(original_symbols ~= detected_symbols) / length(original_symbols); % SER
end
```