CIE 237 Course Project Report

Performance of Correlation Receivers for M-PAM over AWGN Channels

Group 12

Hagar Alsherbiny Atallah 202301124

Yara Mahmoud abdelrazek 202300177

1- Introduction:

Recovering signals attenuated by noise is a key problem in digital communication systems. It is well known that the matching filter, also known as the correlation receiver, performs best in additive white Gaussian noise (AWGN) channels. Using MATLAB and Simulink simulations, this research examines correlation receiver performance for M-level Pulse Amplitude Modulation (PAM) over AWGN. For 2-, 4-, and 8-PAM systems, it specifically examines the relationship between bit error rate (BER) and symbol error rate (SER) and signal-to-noise ratio (SNR).

2- Theory:

2.1 Correlation Receiver:

A correlation receiver compares the received signal against expected waveforms and selects the closest match using a correlation metric. For PAM signals, the receiver correlates the noisy signal with symbol templates. The decision is based on the maximum correlation output, ideally recovering the transmitted symbol. This happens due to this process:

First, we have a signal x(t) and a gaussian noise w(t) and we pass them through a line time invariant filter that separates the noise from the signal, so we have y(t)= h(t)+ xo(t) as seen in figure 1.

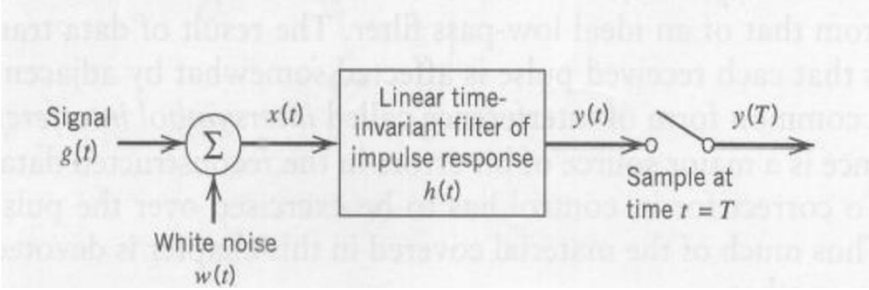


Figure 1

Second, we make sure that the filter makes the power of the output signal at time equal T is greater than the average power of the noise. Therefore, we have a maximum signal to noise ratio so we can easily separate signal from noise.

2.2 M-PAM Modulation

M-PAM maps $\log_2(M)$ bits to one of M amplitude levels. The average symbol energy increases with M, making higher-order PAM more susceptible to noise since having larger M means the spacings between signals is smaller, which also means that their sensitivity to noise is higher, it can be easily mistaken for a neighbor bit. That also means we need higher signal power to have the same performance error wise as the lower M as we know that SNR= P(signal)/ P(noise). That's why, higher M is worse in terms of BER and SER.

2.3 Bit and Symbol Error Rates

- **BER:** The rate at which received bits differs from the transmitted bits.
- **SER:** The rate at which symbols are incorrectly decoded.

Theoretical BER for M-PAM over AWGN:

- For M = 2 (BPSK):

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

- For M > 2:

$$\mathrm{BER}_{\text{IMDD-PAM}} = \frac{N-1}{N\log_2(N)}\operatorname{erfc}\left(\sqrt{\frac{3\log_2(N)}{2(N-1)(2N-1)}\cdot\frac{E_b}{N_0}}\right)$$

3- Methodology:

MATLAB:

3.1 Simulation Parameters:

- **Number of symbols:** 100,000
- **PAM levels:** 2, 4, 8
- **SNR range:** 0 dB to 20 dB (step 2 dB)
- **Channel:** AWGN

3.2 Simulation steps:

- Initiating the signals to be drawn (BER (THEORITICAL, DETECTED), SER):

```
ber_measured = zeros(length(mod_orders), length(snr_range_db));
ber_expected = zeros(length(mod_orders), length(snr_range_db));
ser_measured = zeros(length(mod_orders), length(snr_range_db));
```

- Starting a loop an the different values of M (2,4,8) to calculate the number of bits and energy in each one of them:

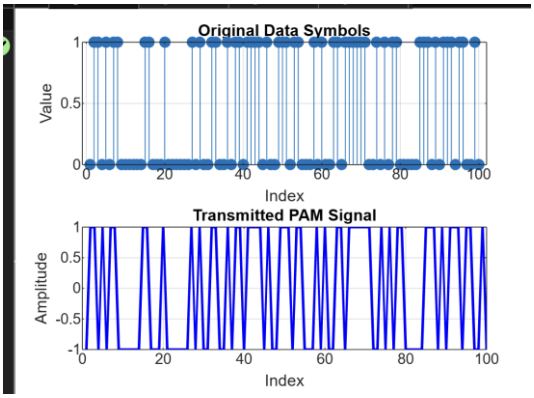
```
for mod_idx = 1:length(mod_orders)
    M = mod_orders(mod_idx);
    bits_per_sym = log2(M);
    energy_sym = (M^2 - 1) / 3;
```

- Inside that loop, for each value of M we generate a random signal and use PAM on it:

```
symbol_stream = randi([0 M-1], num_syms, 1);
tx_signal = pammod(symbol_stream, M);
```

-To know how the signal look modulated and original in 2-PAM:

```
if mod_idx == 1
    figure;
    subplot(2,1,1);
    stem(symbol_stream(1:samples_to_display), 'filled');
    title('Original Data Symbols');
    xlabel('Index'); ylabel('Value'); grid on;

    subplot(2,1,2);
    plot(tx_signal(1:samples_to_display), 'b', 'LineWidth', 1.5);
    title('Transmitted PAM Signal');
    xlabel('Index'); ylabel('Amplitude'); grid on;
end
```



- Starting another loop for the SNR range we have, we use the number of bits for each M and convert the db range to linear so we can calculate the energy for each bit in each symbol and then calculate their variance:

```
for snr_idx = 1:length(snr_range_db)
    snr_db = snr_range_db(snr_idx);
    snr_linear = 10^(snr_db / 10);
    eb_n0 = snr_linear / bits_per_sym;
    noise_var = energy_sym / (2 * eb_n0);
```

When we have high variance, that means EbN0 is higher (higher SNR) so we have less noise and vice versa

- Generating a random gaussian noise using the variance we calculated previously and then adding it to the random signal:

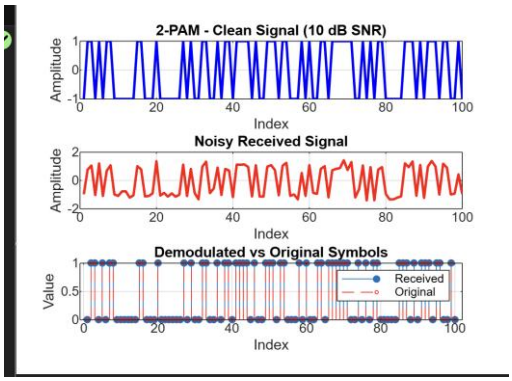
```
awgn = sqrt(noise_var) * randn(num_syms, 1);
rx_signal = tx_signal + awgn;
```

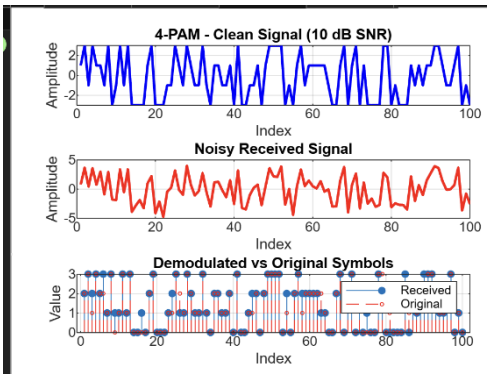
-For each m, we plot the modulated, noise, and the after demodulation (second step), we also plot the original vs the demodulated:

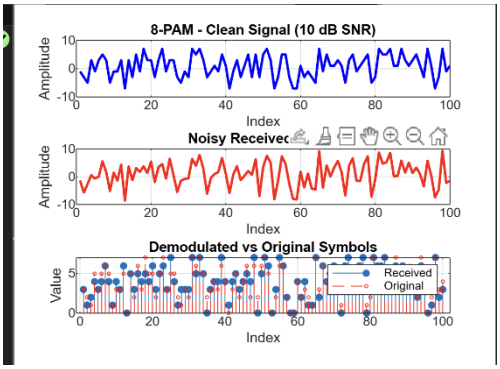
```
if snr_db == 10
    figure;
    subplot(3,1,1);
    plot(tx_signal(1:samples_to_display), 'b', 'LineWidth', 1.5);
    title(sprintf('%d-PAM - Clean Signal (10 dB SNR)', M));
    xlabel('Index'); ylabel('Amplitude'); grid on;

    subplot(3,1,2);
    plot(rx_signal(1:samples_to_display), 'r', 'LineWidth', 1.5);
    title('Noisy Received Signal'); xlabel('Index');
    ylabel('Amplitude'); grid on;
end
```

```
    if snr_db == 10
        subplot(3,1,3);
        stem(sym_received(1:samples_to_display), 'filled', 'MarkerSize', 4);
        hold on;
        stem(symbol_stream(1:samples_to_display), 'r--', 'MarkerSize', 2);
        title('Demodulated vs Original Symbols');
        legend('Received', 'Original');
        xlabel('Index'); ylabel('Value'); grid on;
    end
```

Plots:







-we use inverse PAM to have the detected signal again:

```
sym_received = pamdemod(rx_signal, M);
```

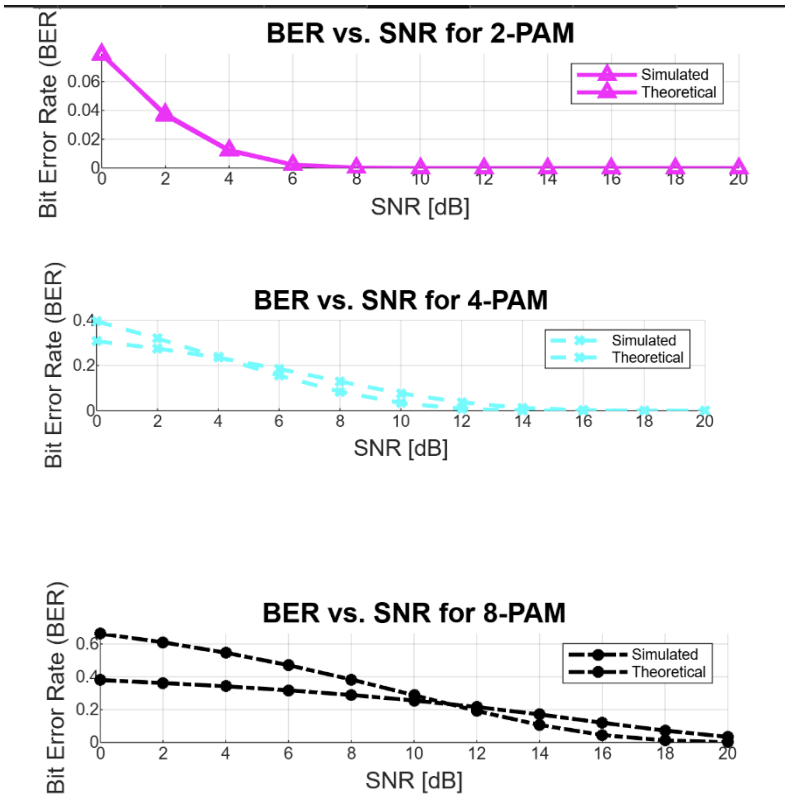
-calculating the number of symbols that was received differently than the data and dividing them over N which is the number of symbols of original random data to have the ratio between them. Also, converting both the data and the detected symbol into bits and comparing between them to have the ratio of error in the bits:

```
ser_measured(mod_idx, snr_idx) = mean(sym_received ~= symbol_stream);

bits_tx = de2bi(symbol_stream, bits_per_sym, 'left-msb');
bits_rx = de2bi(sym_received, bits_per_sym, 'left-msb');
total_bit_errors = sum(bits_tx ~= bits_rx, 'all');
ber_measured(mod_idx, snr_idx) = total_bit_errors / (num_syms * bits_per_sym);
```

-Calculating the theoretical values of BER using the rules we discussed previously:

```
    if M == 2
        ber_expected(mod_idx, snr_idx) = qfunc(sqrt(2 * eb_n0));
    else
        ber_expected(mod_idx, snr_idx) = ...
            2 * (M - 1) / M * qfunc(sqrt((6 * log2(M) / (M^2 - 1)) * eb_n0));
    end
```

-finally plotting theoretical and simulated BER for each M:



-Tables for each SNR vs SER for each M were printed:

```
fprintf('\n--- SYMBOL ERROR RATE REPORT ---\n');
for mod_idx = 1:length(mod_orders)
    fprintf('\nM = %d-PAM\n', mod_orders(mod_idx));
    fprintf('SNR (dB) | SER (Simulated)\n');
    for snr_idx = 1:length(snr_range_db)
        fprintf('%8.2f | %.5f\n', snr_range_db(snr_idx), ser_measured(mod_idx, snr_idx));
    end
end
```

```
M = 2-PAM
SNR (dB) | SER (Simulated)
    0.00 | 0.07773
    2.00 | 0.03693
    4.00 | 0.01233
    6.00 | 0.00267
    8.00 | 0.00023
   10.00 | 0.00000
   12.00 | 0.00000
   14.00 | 0.00000
   16.00 | 0.00000
   18.00 | 0.00000
   20.00 | 0.00000
```

```
M = 4-PAM
SNR (dB) | SER (Simulated)
    0.00 | 0.49294
    2.00 | 0.42876
    4.00 | 0.35846
    6.00 | 0.27857
    8.00 | 0.19455
   10.00 | 0.11785
   12.00 | 0.05658
   14.00 | 0.01912
   16.00 | 0.00359
   18.00 | 0.00025
   20.00 | 0.00000
```

```
M = 8-PAM
SNR (dB) | SER (Simulated)
    0.00 | 0.75237
    2.00 | 0.72131
    4.00 | 0.68299
    6.00 | 0.63440
    8.00 | 0.57417
   10.00 | 0.49983
   12.00 | 0.42080
   14.00 | 0.32417
   16.00 | 0.23009
   18.00 | 0.13719
   20.00 | 0.06530
```

4. Results and Discussion:

From the plots and the tables shown in the previous step:

Each subplot shows the BER performance for a different M-PAM level. Key observations:

- **2-PAM:** Performs best; lowest BER across all SNRs.
- **4-PAM & 8-PAM:** Higher error rates due to closer symbol spacing, especially at low SNRs.
- The simulated BER closely matches the theoretical predictions, validating the correctness of the correlation receiver.
- As expected, SER decreases with increasing SNR.
- Higher M values result in higher SER at the same SNR due to denser constellations.

5. Comparison with Simulink:

MATLAB Code

```
% Parameters
EbNoVec = 0:2:20;  % Eb/No range (dB)
M = 2;             % PAM-2 (Binary PAM)
numBits = 1e6;     % Number of bits to transmit (larger for better accuracy)
simBER = zeros(length(EbNoVec), 1);  % Simulated BER
theoreticalBER = zeros(length(EbNoVec), 1);  % Theoretical BER
```

This defines a **range of Eb/No values (in dB)** from 0 to 20, in steps of 2.
Eb/No is the ratio of bit energy to noise power spectral density — a common measure in digital communications.

This sets **M = 2**, which means we're using **2-level PAM (Binary PAM).**

Simulates **1 million bits** to ensure accurate results (larger sample = better estimate).

Preallocates vectors for storing simulated and theoretical BER values for each Eb/No

```
% Loop over each Eb/No value
for i = 1:length(EbNoVec)
    EbNo = EbNoVec(i);  % Current Eb/No value

    % Generate random bit stream
    transmittedBits = randi([0 M-1], numBits, 1);  % Random bits (0 or 1 for PAM-2)

    % Modulate: PAM-2 (Map bits to 2 symbols, e.g., 0 -> -1, 1 -> 1)
    transmittedSymbols = 2 * transmittedBits - 1;
```

Maps bits to **PAM-2 symbols**:

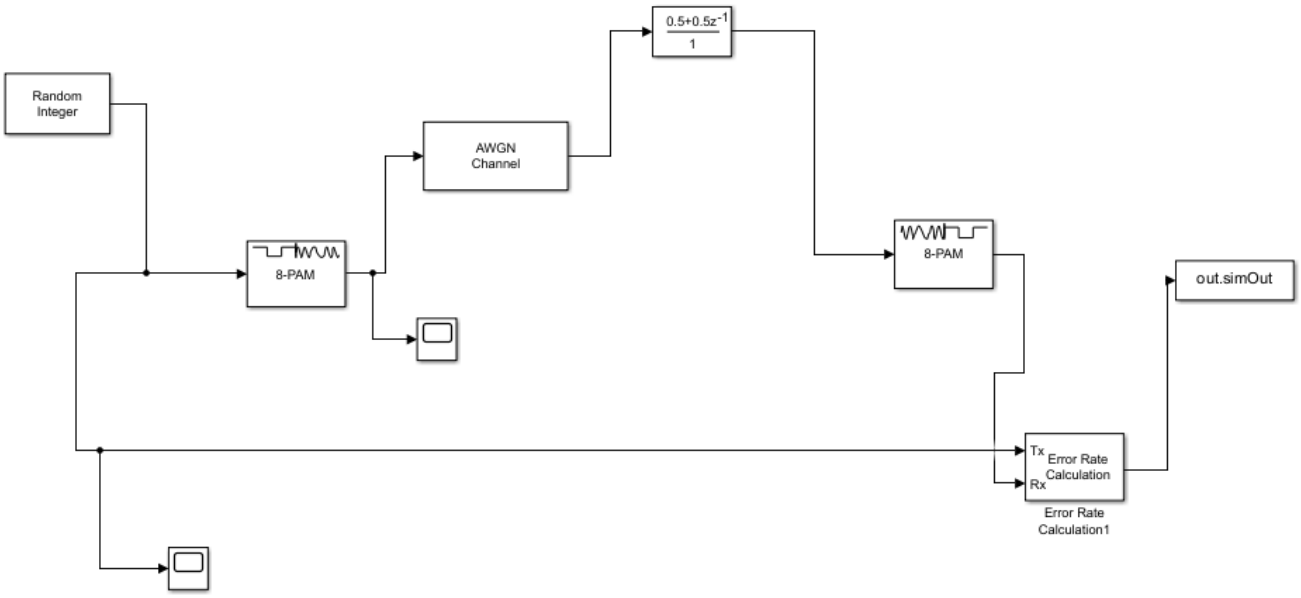
- Bit 0 → Symbol -1
- Bit 1 → Symbol +1

This ensures average signal power is 1.

```
    % Calculate SNR in linear scale (SNR = Eb/No for PAM-2)
    SNR = 10^(EbNo / 10);  % Convert Eb/No to linear SNR

    % Noise variance adjusted based on SNR (signal power is 1 for PAM-2)
    noiseVariance = 1 / (2 * SNR);  % Noise variance for the SNR in linear scale
    noise = sqrt(noiseVariance) * randn(numBits, 1);  % Add Gaussian noise

    % Received symbols with noise
    receivedSymbols = transmittedSymbols + noise;

    % Demodulate: Decision based on received symbols
    receivedBits = receivedSymbols > 0;  % Threshold decision (0 or 1)

    % Compute the BER (Bit Error Rate)
    numErrors = sum(receivedBits ~= transmittedBits);  % Count bit errors
    simBER(i) = numErrors / numBits;  % Simulated BER
```

Compares original and received bits to count **bit errors**.
Calculates **simulated BER** by dividing errors by total bits.

```
    % Theoretical BER using Q-function for 2-PAM
    theoreticalBER(i) = qfunc(sqrt(2 * SNR));  % Q-function for PAM-2

    % Print progress (Optional)
    fprintf('Simulating for Eb/No = %.1f dB, Simulated BER = %.5f, Theoretical BER = %.5f\n', EbNo,
simBER(i), theoreticalBER(i));
end

% Plot the BER vs Eb/No
figure;
semilogy(EbNoVec, simBER, '-o', 'DisplayName', 'Simulated BER'); % Simulated BER
hold on;
semilogy(EbNoVec, theoreticalBER, '-x', 'DisplayName', 'Theoretical BER'); % Theoretical BER
xlabel('Eb/No (dB)');
ylabel('BER');
title('BER vs Eb/No for 2-PAM');
legend('show');
grid on;
```



Random Integer Source:

- **Generates a sequence of random integers that represent the digital data to be transmitted.**

8-PAM Modulator:

- **Converts the digital input into an 8-PAM modulated signal, which consists of 8 distinct amplitude levels.**

AWGN Channel:

- **Adds Gaussian noise to the modulated signal to simulate real-world channel noise, allowing for analysis of signal robustness.**

FIR filter :

To match pulse shape , and maximize output signal.

8-PAM Demodulator:

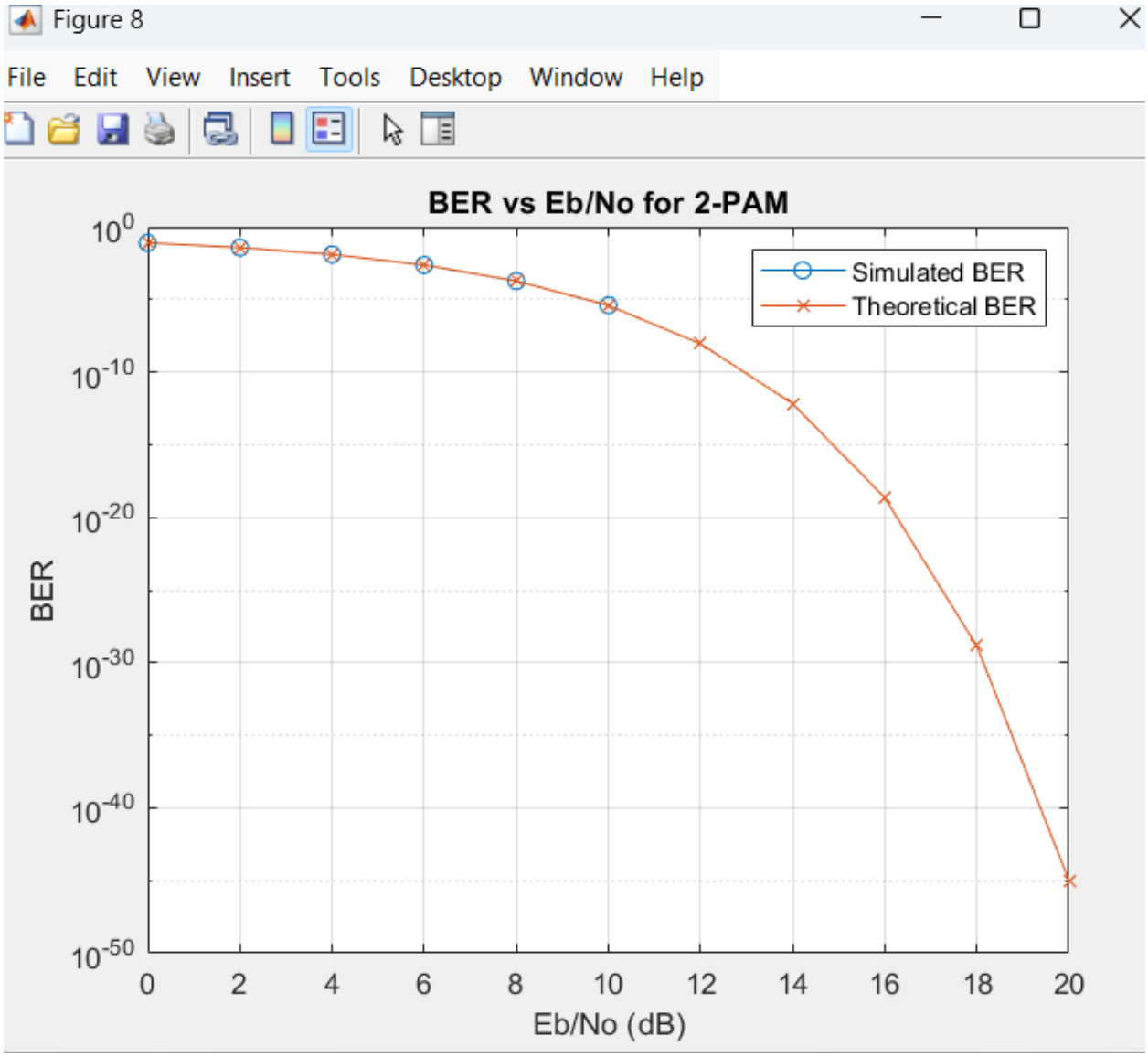
- **Converts the received noisy signal back to a digital format by mapping the received amplitudes to the nearest valid amplitude level.**

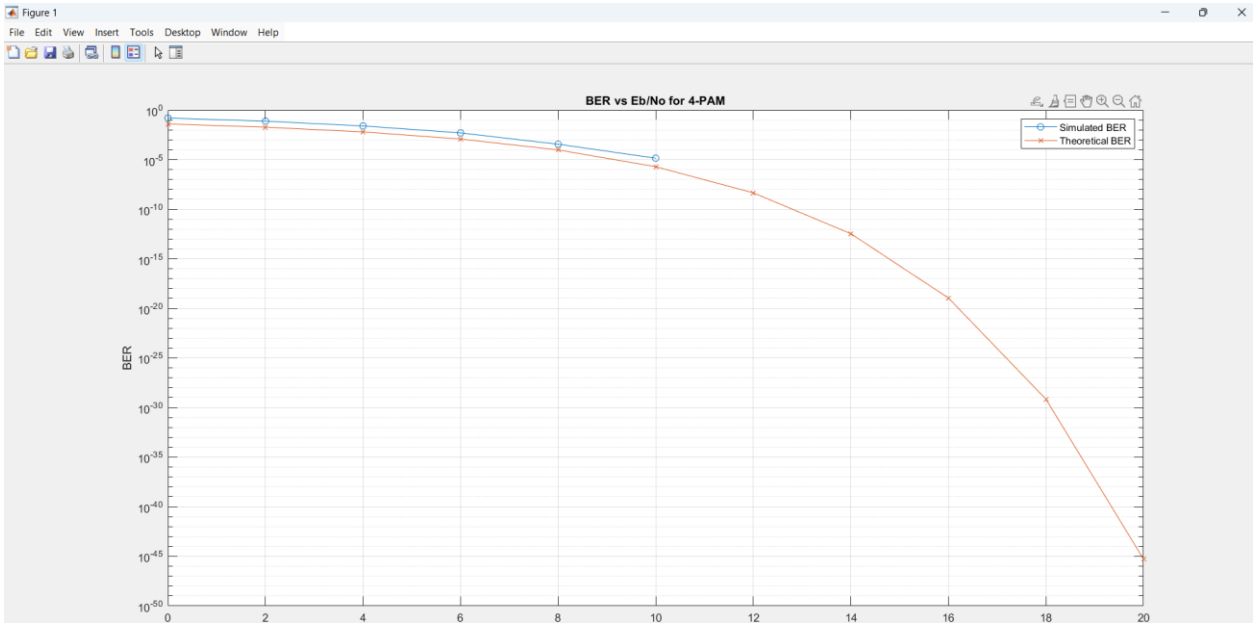
Error Rate Calculation:

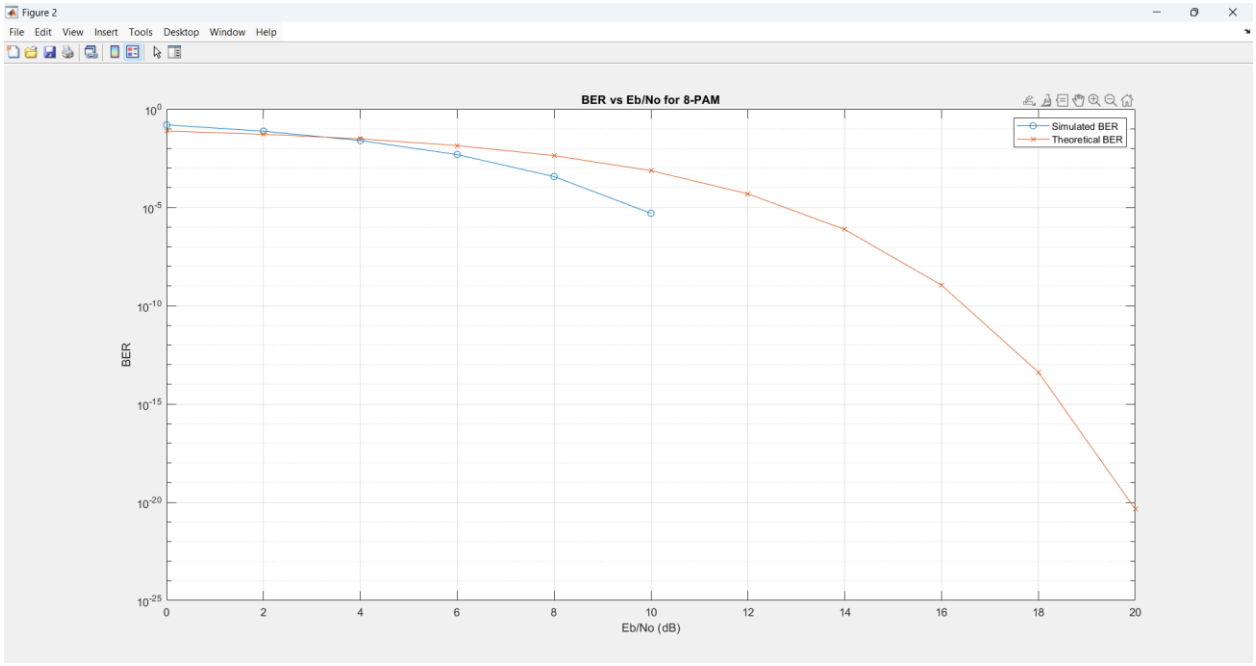
- **Compares the original transmitted data with the received data to calculate the bit error rate (BER).**

Output (out.simOut):

- **Collects the simulation output data for further analysis or export.**

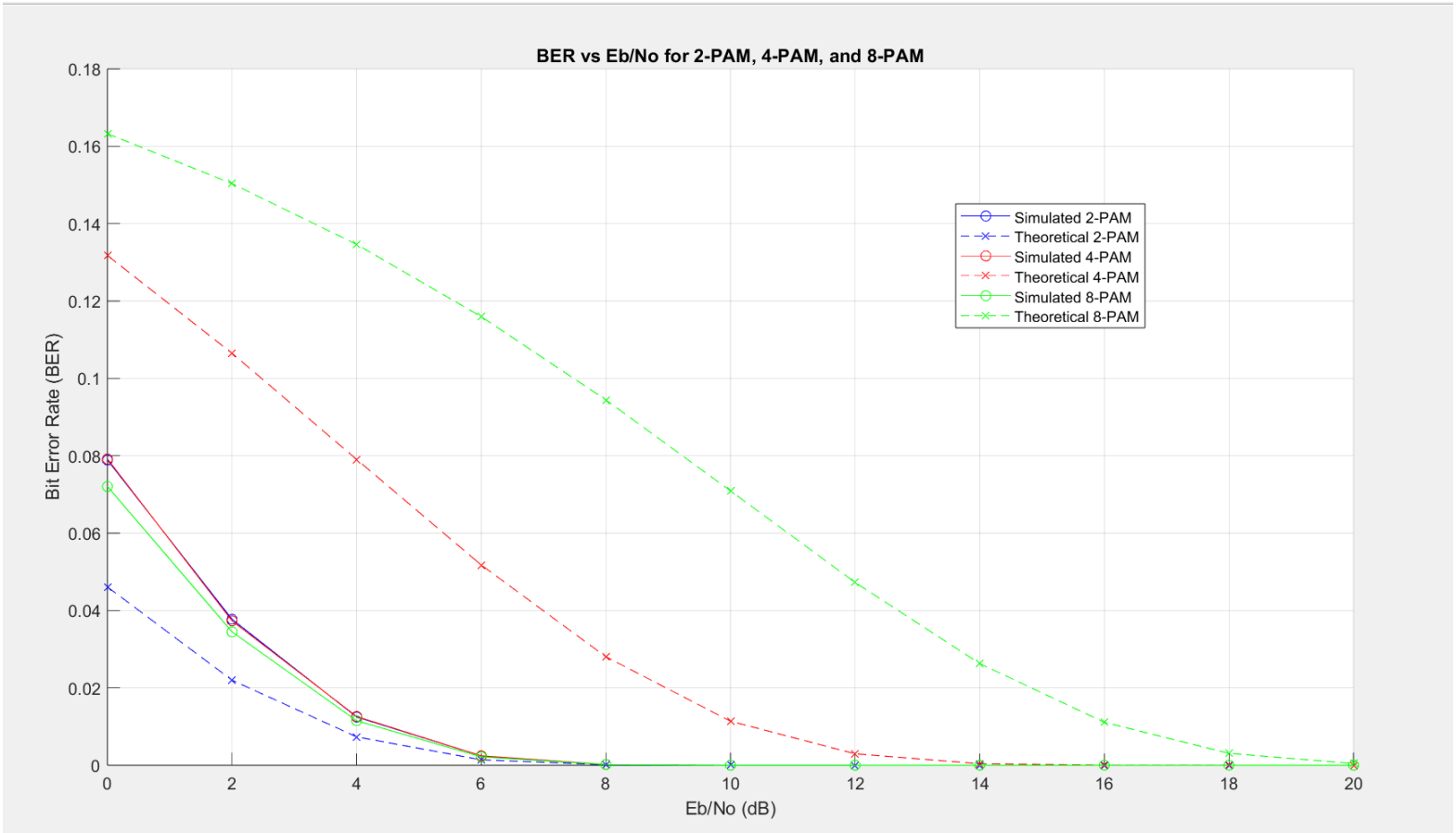






This plot shows the Bit Error Rate (BER) vs. Eb/N0E for a 2-PAM modulation scheme.

- **X-Axis (Eb/No in dB):** Represents the ratio of energy per bit to noise power spectral density, expressed in decibels. Higher values indicate a stronger signal relative to noise.
- **Y-Axis (BER):** Shows the bit error rate on a logarithmic scale. Lower BER values indicate better performance.
- **Simulated BER (Blue Circles):** This is the BER obtained through the simulation of the 2-PAM system over the AWGN channel.
- **Theoretical BER (Orange Line):** The expected BER based on the theoretical analysis of 2-PAM under AWGN conditions.



6. Conclusion:

This project demonstrated the use of correlation receivers for detecting M-PAM signals over AWGN. The simulated BER and SER align well with theoretical expectations. As M increases, the system becomes more susceptible to noise. MATLAB provides a powerful environment to model and analyze such systems.

7. References:

1. Proakis, J.G., *Digital Communications*, McGraw-Hill.
2. MATLAB Documentation: pammod, pamdemod, qfunc