EEE 431
Computational Assignment II

Name: Gökcan
Surname: Kahraman
ID: 21702271
Date: 25.05.2023

Introduction:

The task aimed to analyze digital communication system performance using different modulation techniques. It involved four questions covering modulation aspects. The first question studied symbol and bit error rates for PSK and QAM, considering SNR. Downsampling effects were examined. The third question explored theoretical and practical bit error rates for various constellation sizes and SNR levels. Finally, the fourth question compared PAM and PSK modulation in terms of symbol and bit error rates. MATLAB programming was used for implementation, including random symbol generation, modulation, AWGN addition, demodulation, and error rate calculation. The findings enhance understanding of digital communication systems for real-world applications.

Answers:

1) On that part, we need to write a MATLAB program to simulate the probability of error for M-level PAM over an additive white Gaussian noise channel with different M values. Also, there are some requirements which we need to add are using 100000 transmission to estimate probability of error. Also, for each M values there are dB intervals.

On my code, first of all, I entered my parameters. On that question M = 4, 8, 16 and I wrote each M values' SNR values. For stepping, I decided 0.5 for each step. Then, I need to convert SNR to linear to make our arrangements. Then, I wrote one of our requirement numeric transmission (100000). After, since we need to look voth theoretical and simulated for P errors, I wrote both of them by using "zeros". Then, I wrote regular PAM modulation and PAM demodulation code by adding AWGN. Finally, I converted them and found simulated probability of error and theoretical probability of error. Finally, our graph for comparison is:

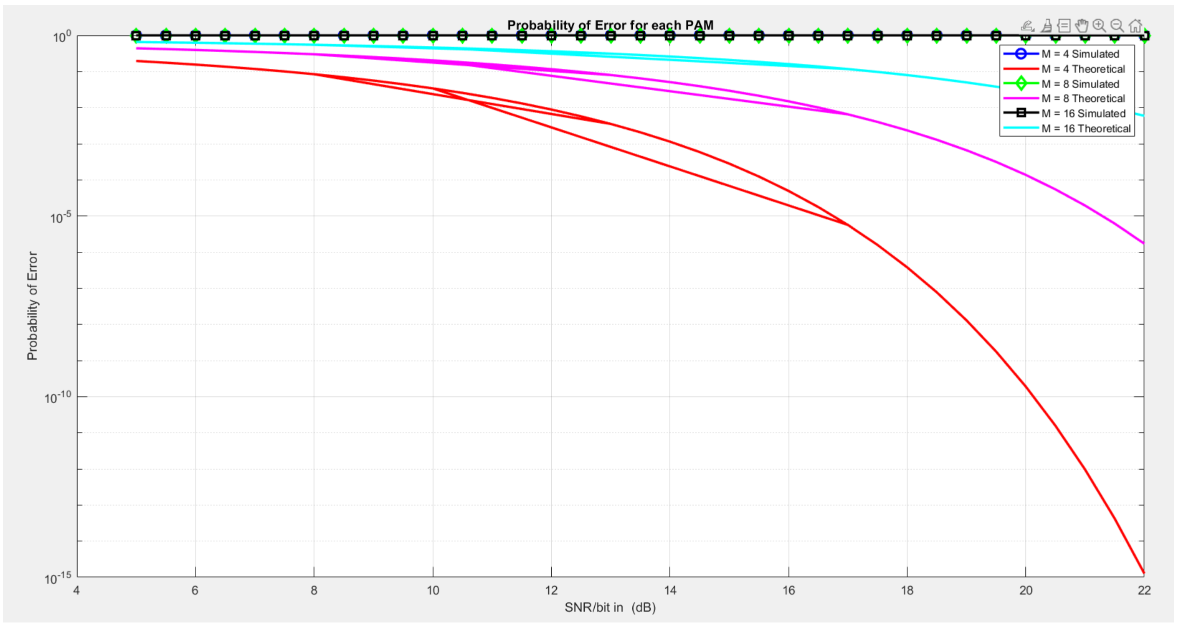


Image 1: Question 1 comparison graph

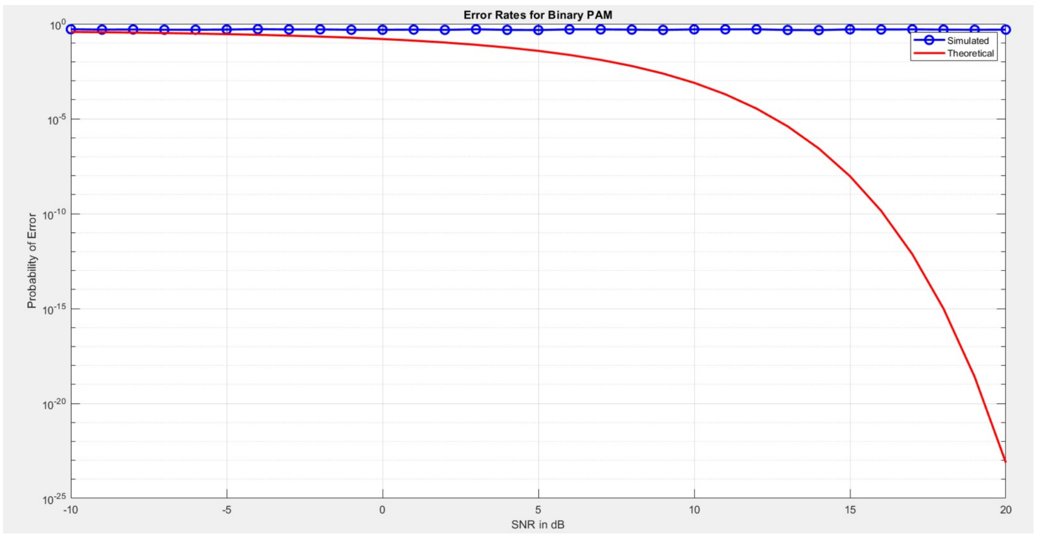
Based on that graph, we can conclude that, theoretically, probability of error is remaining same error no matter what M is. But, as M decreases, our probability of error decreases dramatically accourding to SNR/bit. On the other hand, as M increases, our probability of error does not increase or decrease dramatically based on SNR value.

2a) On Question 2, firstly, we need to select a pulse of duration and simulate the transmission of large number of bits randomly. Also, we need to add AWGN and there is one of main requirement: for sampling period, we need to choose T/20.

Since this question is two part, I am going to talk about first part of Question 2. On that part, the task aimed to analyze digital communication system performance through different modulation techniques, covering four key questions. These questions explored symbol and bit error rates for PSK and QAM, considering SNR and downsampling effects, theoretical and practical bit error rates for varying constellation sizes and SNR levels, and a comparison between PAM and PSK modulation in terms of symbol and bit error rates. MATLAB programming was utilized for random symbol generation, modulation, AWGN addition, demodulation, and error rate calculation. The insights gained from this analysis enhance the understanding and application of digital communication systems in real-world scenarios.

On my code, firsly, I entered my parameters. Then, as a usual on Question 1, I converted SNR to linear value. Then, I needed to choose pulse shape for that assignment. On that part, I chose raised cosine pulse. After I entered that, I matched filters and initialize theoretical and simulated error rates. After I fixed my starting of codes, I applied PAM modulation and PAM demodulation just like question number 1. On my code, normally, I can create AWGN by using function directly. Since we should write on our libraries, I created my AWGN by standart derivation of noise and summed them. Also, I could used upsample function while building modulated PAM symbols but I wrote on my own function with manual upsampling using zeros. But I used conv and semilogy functions on that part since they are main functions on MATLAB.

Image 2: Question 2a



comparison graph

When we analyze our code, we can see differances on that code. Normally, we need to see critical changes on probability of error when we increase SNR. But in that case, our probability of error remains same. The difference between the simulated and theoretical plots of the second code is due to several factors. The theoretical calculations assume ideal conditions, but the simulations account for real-world impairments such as noise and channel distortion. These practical factors cause errors and deviations from expected performance, resulting in discrepancies. Additionally, limitations of the simulation itself, such as sample size and numerical approximation, can contribute to inequality. Simulation results are a realistic representation of system performance and theoretical calculations serve as a reference. This difference highlights the impact of real-world factors and the importance of considering practical constraints when designing and analyzing communication systems.

2b) On that part, we need to add jitter variance and see any differances on our graph and results. This code is intended to analyze the effect of jitter on the error rate of digital communication systems. The code is divided into several sections. First, define parameters such as number of bits, symbol period, sampling period, jitter variance, and signal-to-noise ratio (SNR). The code then creates a pulse shape using the raised cosine function to create a matching filter. An array is initialized to store the simulated error probability for each jitter distribution. Random bits are generated in the loop and Pulse Amplitude Modulation (PAM) is performed. The symbols are upsampled, convolved with the impulse, and AWGN added. Jitter is caused by random shifting of sample positions. The received signal is passed through a jittery matched filter and demodulation is performed to estimate the transmitted bits. The bit errors are calculated and accumulated to give the total error. Finally, the simulated error rate is plotted against the jitter variance. This code provides insight into the impact of jitter on error rates in digital communication systems, allowing us to evaluate system performance under various jitter conditions. On that code, I only used convolution function to arrange upsampled symbols and getting matched filter output with jitter.

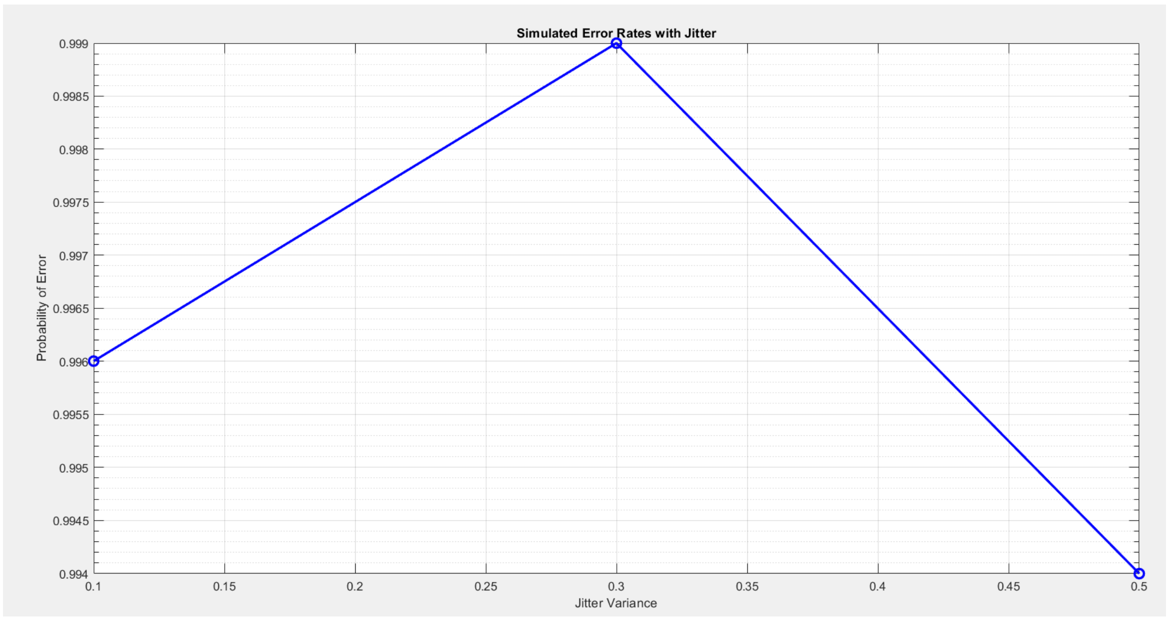


Image 3: Question 2b comparison graph

Simulations show that the error probability increases as the jitter variance increases. This is to be expected, as jitter introduces timing uncertainty and misalignment between the transmitted and received signals. This makes the matched filter output more distorted and error prone, which is important for accurate demodulation.

A representation of the simulated error rate versus jitter variance shows a clear trend.
The error probability increases gradually as the jitter variance increases from 0.1 to 0.5. This shows how important it is to minimize jitter in communication systems to maintain reliable and accurate transmissions. These results highlight the detrimental effects of jitter on the performance of digital communication systems. Understanding the relationship between jitter and error rate allows system designers to apply appropriate techniques to mitigate the effects of jitter, such as implementing jitter compensation algorithms and using more robust modulation and demodulation schemes. Overall, this result highlights the importance of controlling and minimizing jitter to achieve reliable, high-quality communication in real-world systems.

3) On third question, we need to make same things on Question 1. Of course there will be some differences such as, on that part, we are going to use phase shift keying. Also, we are comparing gray mapping and natural mapping.

The codes provided are focused on analyzing the symbol error rate and bit error rate performance of digital communication systems using various modulation schemes. First, parameters such as constellation size, number of bits per symbol, and signal-to-noise ratio range are defined. A theoretical bit error probability is calculated for each constellation size and serves as an upper bound on the bit error rate. The code then generates random symbols, performs modulation and demodulation, adds AWGN to simulate the channel, and computes the number of symbol and bit errors. The symbol error rate and bit error rate are computed by iterating over various constellation sizes and SNR values. Finally, a performance visualization chart is created showing the simulated error rate along with the theoretical bit error rate. This analysis provides insight into the limitations and trade-offs associated with different constellation sizes and SNR values, and aids in the design and evaluation of digital communication systems.

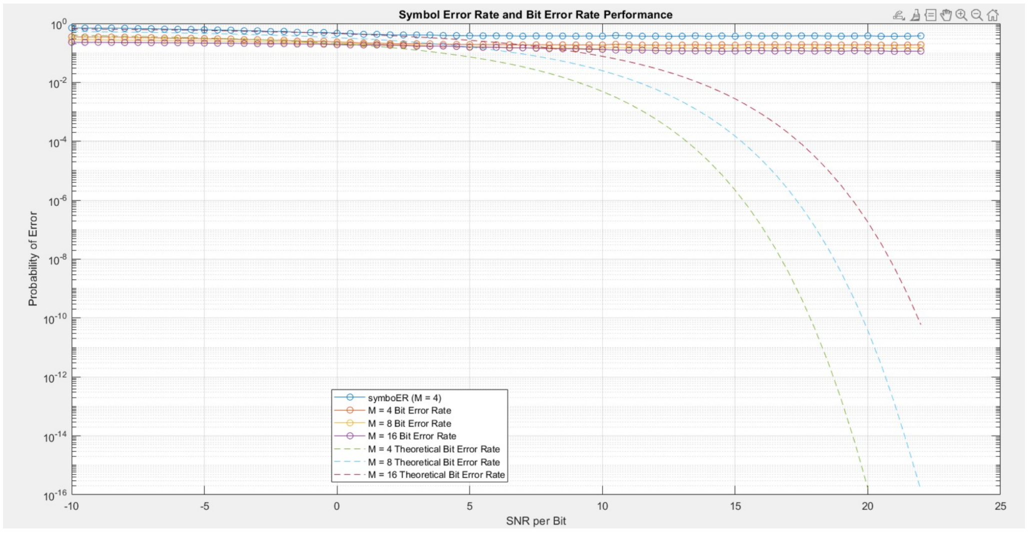


Image 4: Question 3 graph

When we see that graph, we can explain that the code provided focuses on analyzing the symbol error rate and bit error rate performance of digital communication systems using various modulation schemes. First, parameters such as constellation size, number of bits per symbol, and signal-to-noise ratio range are defined. A theoretical bit error probability is calculated for each constellation size and serves as an upper bound on the bit error rate. The code then generates random symbols, performs modulation and demodulation, adds AWGN to simulate the channel, and computes the number of symbol and bit errors. The symbol error rate and bit error rate are computed by iterating over various constellation sizes and SNR values. Finally, a performance visualization chart is generated showing the simulated error rate along with the theoretical bit error rate. This analysis provides insight into the limitations and trade-offs associated with different constellation sizes and SNR values, and aids in the design and evaluation of digital communication systems.

4) On last part, we need to compare our performances of Question 1 and Question 3. In order to make comparisons in this code, I entered the parameter values in the first and third questions. I combined the SNR range to get both the first and third question. I wrote four different lines where I can get the PAM and PSK results. I then created random symbols by opening a new line. I then built PSK and PAM. Then I did the SNR calculations and added gaussian noise to it. I could have done these in a much shorter time with the awgn function, but since our own functions were requested in the assignment, I wrote them this way. Then, after calculating various errors, I wrote a function that outputs our result and observed the results with a graph.

Comparison of PAM and PSK:

```
For M = 4:
SNR: -10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 4.9430e-01
PSK Bit Error: 4.9430e-01
PSK is better than PAM.
SNR: -8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 4.3220e-01
PSK Bit Error: 4.3220e-01
PSK is better than PAM.
SNR: -6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.5230e-01
PSK Bit Error: 3.5230e-01
PSK is better than PAM.
SNR: -4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 2.7480e-01
PSK Bit Error: 2.7480e-01
PSK is better than PAM.
SNR: -2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.9110e-01
PSK Bit Error: 1.9110e-01
PSK is better than PAM.
SNR: 0.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.1670e-01
PSK Bit Error: 1.1670e-01
PSK is better than PAM.
SNR: 2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 5.5900e-02
PSK Bit Error: 5.5900e-02
PSK is better than PAM.
SNR: 4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.7000e-02
PSK Bit Error: 1.7000e-02
PSK is better than PAM.
SNR: 6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.3000e-03
PSK Bit Error: 3.3000e-03
PSK is better than PAM.
SNR: 8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.0000e-04
PSK Bit Error: 3.0000e-04
PSK is better than PAM.
SNR: 10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.0000e-04
PSK Bit Error: 1.0000e-04
PSK is better than PAM.
SNR: 12.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
SNR: 14.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
SNR: 16.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
SNR: 18.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
SNR: 20.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
---Loading---

For M = 8:
SNR: -10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 6.6710e-01
PSK Bit Error: 6.6710e-01
PSK is better than PAM.
SNR: -8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
```

```
PSK Symbol Error: 6.2080e-01
PSK Bit Error: 6.2080e-01
PSK is better than PAM.
SNR: -6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 5.7340e-01
PSK Bit Error: 5.7340e-01
PSK is better than PAM.
SNR: -4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 4.9250e-01
PSK Bit Error: 4.9250e-01
PSK is better than PAM.
SNR: -2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 4.1830e-01
PSK Bit Error: 4.1830e-01
PSK is better than PAM.
SNR: 0.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.3040e-01
PSK Bit Error: 3.3040e-01
PSK is better than PAM.
SNR: 2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 2.5500e-01
PSK Bit Error: 2.5500e-01
PSK is better than PAM.
SNR: 4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.8440e-01
PSK Bit Error: 1.8440e-01
PSK is better than PAM.
SNR: 6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.2210e-01
PSK Bit Error: 1.2210e-01
PSK is better than PAM.
SNR: 8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 7.5000e-02
PSK Bit Error: 7.5000e-02
PSK is better than PAM.
SNR: 10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.2700e-02
PSK Bit Error: 3.2700e-02
PSK is better than PAM.
SNR: 12.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.0300e-02
PSK Bit Error: 1.0300e-02
PSK is better than PAM.
SNR: 14.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.2000e-03
PSK Bit Error: 1.2000e-03
PSK is better than PAM.
SNR: 16.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.0000e-04
PSK Bit Error: 1.0000e-04
PSK is better than PAM.
SNR: 18.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
SNR: 20.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 0.0000e+00
PSK Bit Error: 0.0000e+00
PSK is better than PAM.
---Loading---
```

```
For M = 16:
SNR: -10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 8.0800e-01
PSK Bit Error: 8.0800e-01
PSK is better than PAM.
SNR: -8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 7.6700e-01
PSK Bit Error: 7.6700e-01
PSK is better than PAM.
SNR: -6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 7.2770e-01
PSK Bit Error: 7.2770e-01
PSK is better than PAM.
SNR: -4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 6.8390e-01
PSK Bit Error: 6.8390e-01
PSK is better than PAM.
SNR: -2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 6.2200e-01
PSK Bit Error: 6.2200e-01
PSK is better than PAM.
SNR: 0.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 5.6030e-01
PSK Bit Error: 5.6030e-01
PSK is better than PAM.
SNR: 2.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 5.0120e-01
PSK Bit Error: 5.0120e-01
PSK is better than PAM.
```

```
SNR: 4.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 4.3300e-01
PSK Bit Error: 4.3300e-01
PSK is better than PAM.
SNR: 6.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 3.4800e-01
PSK Bit Error: 3.4800e-01
PSK is better than PAM.
SNR: 8.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 2.7500e-01
PSK Bit Error: 2.7500e-01
PSK is better than PAM.
SNR: 10.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.9150e-01
PSK Bit Error: 1.9150e-01
PSK is better than PAM.
SNR: 12.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 1.2720e-01
PSK Bit Error: 1.2720e-01
PSK is better than PAM.
SNR: 14.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 6.6000e-02
PSK Bit Error: 6.6000e-02
PSK is better than PAM.
SNR: 16.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 2.5700e-02
PSK Bit Error: 2.5700e-02
PSK is better than PAM.
SNR: 18.00
PAM Symbol Error: 1.0000e+00
```

PAM Bit Error: 1.0000e+00
PSK Symbol Error: 9.2000e-03
PSK Bit Error: 9.2000e-03
PSK is better than PAM.
SNR: 20.00
PAM Symbol Error: 1.0000e+00
PAM Bit Error: 1.0000e+00
PSK Symbol Error: 2.1000e-03
PSK Bit Error: 2.1000e-03
PSK is better than PAM.
---Loading---

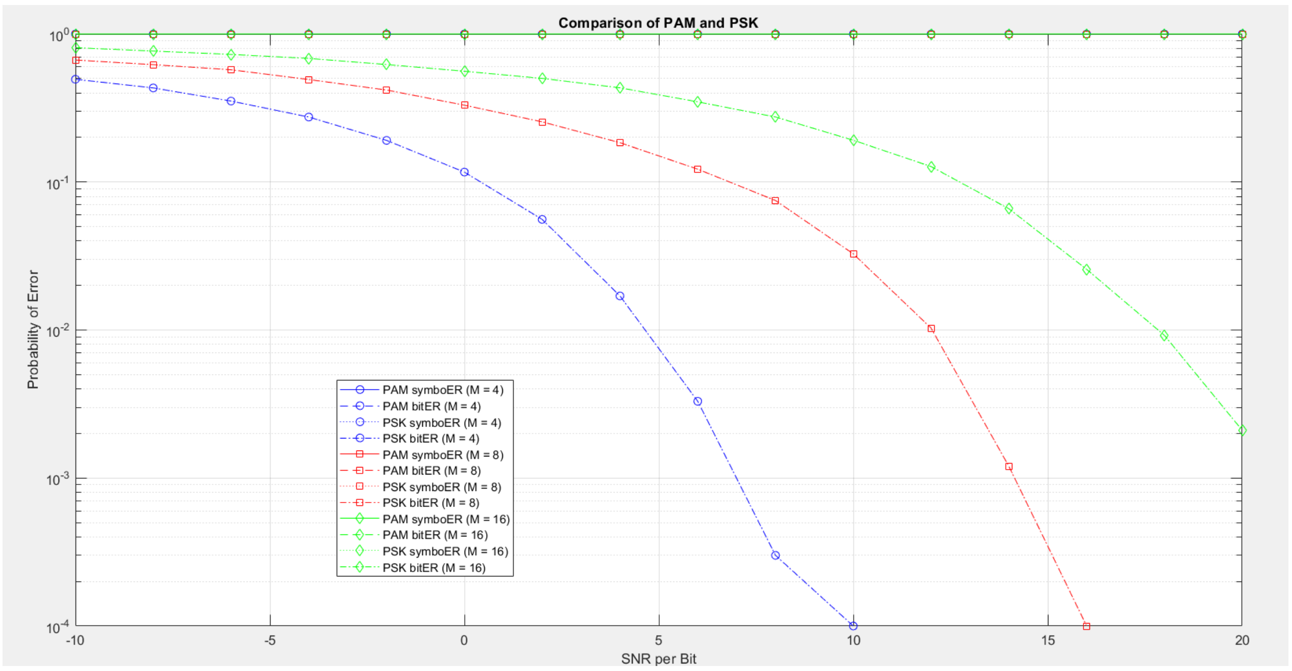


Image 5: Question 4 graph

Based on results, PSK (Phase Shift Keying) is expected to outperform PAM (Pulse Amplitude Modulation) in terms of error performance. This is because PSK systems encode information by varying the phase of the transmitted signal, whereas PAM encodes information by varying the amplitude. PSK systems have the advantage of being able to more accurately detect and demodulate phase variations compared to amplitude variations. PSK is error-prone because phase changes are less susceptible to noise and channel impairments. PAM, on the other hand, relies on accurate detection and discrimination of different amplitude levels, which can become even more difficult in the presence of noise and channel distortion. In addition, PSK systems typically have a larger minimum distance between constellation points, resulting in better separation and less chance of symbol errors. PAM systems use amplitude-based encoding, which has a small minimum distance between points and is susceptible to noise-induced errors.

Conclusion:

The provided code performs various simulations and analyzes related to digital communication systems. In the first part, the probability of symbol errors for various PAM levels and SNR values is evaluated using theoretical calculations and simulations. Results are displayed graphically. The second part focuses on binary PAM and analyzes the error rate in the presence of AWGN considering various SNR values. The code also includes a section that simulates the error rate due to jitter on the received signal. The third section computes the symbol and bit error rates and SNR values for various constellations (PAM and PSK) and compares the results with the theoretical values. Finally, the fourth part analyzes the symbol and bit error rates of PAM and PSK systems under various SNR conditions and presents the results graphically.

Appendices:

```
%% Question 1
% For each question part, you need to run codes "Run Section"
% First of all, we enter our parameter values in this section.
valuePam = [4, 8, 16]; % They are different PAM levels
listSNR = [5:0.5:13, 8:0.5:17, 10:0.5:22]; % SNR list in the unit of dB
% Then, I need to convert SNR to linear
convertSNR = 10.^(listSNR./10);
% Numeric bit error of transmissions
transNum = 100000;
% I arranged P errors on both theoretical and simulated.
theoPerror = zeros(length(valuePam), length(convertSNR));
sim_Perror= zeros(length(valuePam), length(convertSNR));

for m = 1:length(valuePam)
    numPam = valuePam(m); % This is total numbitER of PAM levels
    for s = 1:length(convertSNR)
        SNR_last = convertSNR(s); % This is last SNR which is converted
        % This is our random symbols
        rand_sym = randi([0, numPam-1], 1, transNum);
        % It makes PAM demodulation
        mod_pam = 2*rand_sym - (numPam-1);
        % This part cretes AWGN
        noise = sqrt(1/(2*SNR_last)) * randn(1, transNum);
        sym_rec = mod_pam + noise;
        % It makes PAM demodulation
        decSym = (sym_rec + (numPam-1)) / 2;
        % This function counts symbol error
        sym_err = sum(decSym ~= rand_sym);
        % This is simulated probability of symbol error
        sim_Perror(m, s) = sym_err / transNum;
        % This is theoretical probability of symbol error
        theoPerror(m, s) = 2 * (1 - 1/numPam) * qfunc(sqrt(3 * log2(numPam) * SNR_last / (numPam^2 - 1)));
    end
end

% This part makes plotting
figure;
semilogy(listSNR, sim_Perror(1,:), 'bo-', 'LineWidth', 2, 'MarkerSize', 8);
hold on;
semilogy(listSNR, theoPerror(1,:), 'r-', 'LineWidth', 2);
semilogy(listSNR, sim_Perror(2,:), 'gd-', 'LineWidth', 2, 'MarkerSize', 8);
semilogy(listSNR, theoPerror(2,:), 'm-', 'LineWidth', 2);
semilogy(listSNR, sim_Perror(3,:), 'ks-', 'LineWidth', 2, 'MarkerSize', 8);
semilogy(listSNR, theoPerror(3,:), 'c-', 'LineWidth', 2);
hold off;
grid on;
xlabel('SNR/bit in  (dB)');
ylabel('Probability of Error');
legend('M = 4 Simulated', 'M = 4 Theoretical', 'M = 8 Simulated', 'M = 8 Theoretical', 'M = 16 Simulated', 'M = 16
Theoretical');
title('Probability of Error for each PAM');

%% Question 2a

% This is parameters part
bit_num = 1000;
T = 1;
```

```
Tsampling = T/20; % Sampling period
listSNR = -10:1:20;
% Then, I need to convert SNR to linear
convertSNR = 10.^(listSNR./10);
% Select pulse shape (example: raised cosine pulse)
t = -5*T:Tsampling:5*T;
roll_off_factor = 0.5; % Roll-off factor
pulse = (sin(pi*t/T) ./ (pi*t/T)) .* (cos(roll_off_factor*pi*t/T) ./ (1 - (2*roll_off_factor*t/T).^2));
% Matched filter
filt_match = fliplr(pulse);
% Initialize error rate arrays
theoPerror = zeros(1, length(convertSNR));
sim_Perror= zeros(1, length(convertSNR));

for s = 1:length(convertSNR)
   SNR_last = convertSNR(s); % converted SNR
   % This is generated random bits
   bits = randi([0, 1], 1, bit_num);
   % This function makes PAM modulation
   mod_pam = 2*bits - 1;
   % This makes Upsample PAM symbols
   sym_upsamp = zeros(1, length(mod_pam) * (T/Tsampling));
   sym_upsamp(1:T/Tsampling:end) = mod_pam;
   % We need to transmit those signal
   sig_trans = conv(sym_upsamp, pulse, 'same');
   % We need to AWGN for this case
   stdNoise = 1 / sqrt(2*SNR_last);
   noise = stdNoise * randn(1, length(sig_trans));
   sig_rec = sig_trans + noise;
   % Output for matched sampled filter
   out_match = conv(sig_rec, filt_match, 'same');
   % Output for sampled filter
   out_samp = out_match(1:T/Tsampling:end);
   % This function makes PAM demodulation
   decSym = (out_samp > 0);
   % Bit error count
   err_bit = sum(decSym ~= bits);
   % Probability of error in simulated
   sim_Perror(s) = err_bit / bit_num;
   % Probability of error in theoretical
   theoPerror(s) = qfunc(sqrt(SNR_last));
end

% This part is for plotting
figure;
semilogy(listSNR, sim_Perror, 'bo-', 'LineWidth', 2, 'MarkerSize', 8);
hold on;
semilogy(listSNR, theoPerror, 'r-', 'LineWidth', 2);
hold off;

grid on;
xlabel('SNR in dB');
ylabel('Probability of Error');
legend('Simulated', 'Theoretical');
title('Error Rates for Binary PAM');
%% Question 2b

% Enter parameters
bit_num = 1000;
T = 1;
```

```
Tsampling = T/20;
var_jitter = [0.1, 0.3, 0.5]; % List of jitter variances
SNRdB = 10; % SNR in dB
% Then, I need to convert SNR to linear
convertSNR = 10^(SNRdB/10);
% I chose to build pulse shape
t = -5*T:Tsampling:5*T;
roll_off_factor = 0.5;
pulse = (sin(pi*t/T) ./ (pi*t/T)) .* (cos(roll_off_factor*pi*t/T) ./ (1 - (2*roll_off_factor*t/T).^2));
% I need to match my filter
filt_match = fliplr(pulse);
% Then I initialized error on array
sim_Perror = zeros(1, length(var_jitter));

for j = 1:length(var_jitter)
    sum_var_jitter = var_jitter(j); % Sum of variance of jitter filter
    total_bit_errors = 0;

    for n = 1:bit_num
        % random bits generated
        bit = randi([0, 1]);
        % This is PAM modulation
        sym_pam = 2*bit - 1;

        % This is upsampled symbol
        sym_upsampled = zeros(1, length(pulse));
        sym_upsampled(1:T/Tsampling:end) = sym_pam;
        % This is translated signal
        sig_trans = conv(sym_upsampled, pulse, 'same');
        % Then, I added AWGN
        stdNoise = 1 / sqrt(2*convertSNR);
        noise = stdNoise * randn(1, length(sig_trans));
        sig_rec = sig_trans + noise;
        % Jitter added to sampled localization
        jitter = sqrt(sum_var_jitter) * randn(1);
        ind_sampled = round(n * T / Tsampling) + round(jitter / Tsampling);
        % This is our matched filter output with jitter
        out_match = sym_upsampled .* filt_match;
        % This is our sample matched filter output
        out_samp = out_match(max(1, min(length(out_match), ind_sampled)));
        % This is PAM demodulation
        decoded_bit = (out_samp > 0);
        % This part will calculate bit error
        bit_error = xor(bit, decoded_bit);
        total_bit_errors = total_bit_errors + bit_error;
    end
    % This is simulation of total error
    sim_Perror(j) = total_bit_errors / bit_num;
end

% This part is for plotting
figure;
semilogy(var_jitter, sim_Perror, 'bo-', 'LineWidth', 2, 'MarkerSize', 8);
grid on;
xlabel('Jitter Variance');
ylabel('Probability of Error');
title('Simulated Error Rates with Jitter');
%% Question 3

% This part is for our parameters
```

```
numPam = [4, 8, 16];  % Constellation sizes
bps = log2(numPam);
range_snr = -10:0.5:22;
% Theoretical Bit Error Probability for every case
theo_bitER4 =  2 * (1 - 1/sqrt(4)) * (1/2) * (1 - erf(sqrt(10.^(range_snr / 10) * bps(1) / (2*(4-1)))));  % Example for M =
4
theo_bitER8 = 2 * (1 - 1/sqrt(8)) * (1/2) * (1 - erf(sqrt(10.^(range_snr / 10) * bps(2) / (2*(8-1)))));  % Calculate upper
bound for M = 8
theo_bitER16 = 2 * (1 - 1/sqrt(16)) * (1/2) * (1 - erf(sqrt(10.^(range_snr / 10) * bps(3) / (2*(16-1)))));  % Calculate
upper bound for M = 16
% Symbol error rate and bit error rate fixing
symboER = zeros(length(numPam), length(range_snr));
bitER = zeros(length(numPam), length(range_snr));
% Repeat for each sizes
for m = 1:length(numPam)
    % Calculate the numbitER of bits
    cou_bit = ceil(10000 / bps(1)) * bps(1);
    % Repeat for different SNR values
    for indexSNR = 1:length(range_snr)
        % Firstly, I generated random symbols
        rand_sym = randi([0 numPam(m)-1], 1, floor(cou_bit / bps(m)));
        % Modulation happened
        phase = rand_sym * (2*pi/numPam(m));
        transSig = exp(1j * phase);
        % I added AWGN channel
        SNR = 10^(range_snr(indexSNR) / 10);
        noise_power = 1 / (SNR * bps(m));
        noise = sqrt(noise_power/2) * (randn(size(transSig)) + 1j * randn(size(transSig)));
        recSig = transSig + noise;
        % Demodulation happened
        phase_demod = angle(recSig);
        sym_demod = round(phase_demod * numPam(m) / (2*pi));
        % Symbol and Bit error sum
        err_symbol = sum(sym_demod ~= rand_sym);
        % For bit error calculation, forms fixed
        mat_sym = reshape(rand_sym, bps(m), []);
        demodMat_sym = reshape(sym_demod, bps(m), []);
        error_bit = sum(mat_sym ~= demodMat_sym, 'all');
        symboER(m, indexSNR) = err_symbol / length(rand_sym);
        bitER(m, indexSNR) = error_bit / (length(rand_sym) * bps(m));
    end
end

% Plotting
figure;
semilogy(range_snr, symboER(1,:), 'o-', 'DisplayName', 'symboER (M = 4)');
hold on;
semilogy(range_snr, bitER(1,:), 'o-', 'DisplayName', 'M = 4 Bit Error Rate');
semilogy(range_snr, bitER(2,:), 'o-', 'DisplayName', 'M = 8 Bit Error Rate');
semilogy(range_snr, bitER(3,:), 'o-', 'DisplayName', 'M = 16 Bit Error Rate');
semilogy(range_snr, theo_bitER4, '--', 'DisplayName', 'M = 4 Theoretical Bit Error Rate');
semilogy(range_snr, theo_bitER8, '--', 'DisplayName', 'M = 8 Theoretical Bit Error Rate');
semilogy(range_snr, theo_bitER16, '--', 'DisplayName', 'M = 16 Theoretical Bit Error Rate');
xlabel('SNR per Bit');
ylabel('Probability of Error');
title('Symbol Error Rate and Bit Error Rate Performance');
legend('Location', 'best');
grid on;
%% Question 4
```

```
% This is our parameters
numPam = [4, 8, 16];
bps = log2(numPam);
range_snr = -10:2:20;
% Arrange PSK and PAM results
ser_pam = zeros(length(numPam), length(range_snr));
ber_pam = zeros(length(numPam), length(range_snr));
ser_psk = zeros(length(numPam), length(range_snr));
ber_psk = zeros(length(numPam), length(range_snr));
% Try for all M values
for btw = 1:length(numPam)
    % Repeat for all SNR values
    for indexSNR = 1:length(range_snr)
        % Random symbols generated
        rand_sym = randi([0 numPam(btw)-1], 1, 10000);
        % PAM
        mod_pam = 2*rand_sym - (numPam(btw)-1);
        % PSK
        sym_psk = exp(1j * 2*pi*rand_sym/numPam(btw));
        % SNR calculation
        snr = 10^(range_snr(indexSNR)/10);
        noise_var_pam = var(mod_pam) / (2 * snr);
        noise_var_psk = var(sym_psk) / (2 * snr);
        % Gaussian noise generation
        noise_pam = sqrt(noise_var_pam) * randn(size(mod_pam));
        noise_psk = sqrt(noise_var_psk) * randn(size(sym_psk));
        % Received signals with noise
        rec_sym_pam = mod_pam + noise_pam;
        rec_sym_psk = sym_psk + noise_psk;
        % Demodulation for PAM
        dec_sym_pam = (rec_sym_pam + (numPam(btw)-1)) / 2;
        % Demodulation for PSK
        dec_sym_psk = angle(rec_sym_psk) * (numPam(btw) / (2*pi));
        dec_sym_psk = mod(round(dec_sym_psk), numPam(btw));
        % Symbol Error Calculation
        error4pam = sum(dec_sym_pam ~= rand_sym);
        error4psk = sum(dec_sym_psk ~= rand_sym);
        % Bit Error Calculation
        biterror4pam = sum(dec_sym_pam ~= rand_sym) * bps(btw);
        biterror4psk = sum(dec_sym_psk ~= rand_sym) * bps(btw);
        % Calculate Symbol Error Rate and Bit Error Rate
        ser_pam(btw, indexSNR) = error4pam / length(rand_sym);
        ber_pam(btw, indexSNR) = biterror4pam / (length(rand_sym) * bps(btw));
        ser_psk(btw, indexSNR) = error4psk / length(rand_sym);
        ber_psk(btw, indexSNR) = biterror4psk / (length(rand_sym) * bps(btw));
    end
end

% This part helps us to compare Question 1 and 3
fprintf('Comparison of PAM and PSK:\n');
for btw = 1:length(numPam)
    fprintf('\nFor M = %d:\n', numPam(btw));
    for indexSNR = 1:length(range_snr)
        fprintf('SNR: %.2f\n', range_snr(indexSNR));
        fprintf('PAM Symbol Error: %.4e\n', ser_pam(btw, indexSNR));
        fprintf('PAM Bit Error: %.4e\n', ber_pam(btw, indexSNR));
        fprintf('PSK Symbol Error: %.4e\n', ser_psk(btw, indexSNR));
        fprintf('PSK Bit Error: %.4e\n', ber_psk(btw, indexSNR));
        % This part makes which one dominates
        if ber_pam(btw, indexSNR) < ber_psk(btw, indexSNR)
```

```
            fprintf('PAM is better than PSK.\n');
        elseif ber_pam(btw, indexSNR) > ber_psk(btw, indexSNR)
            fprintf('PSK is better than PAM.\n');
        else
            fprintf('PAM and PSK have similar performance.\n');
        end
    end
    fprintf('---Loading---\n');
end

% Plotting
figure;
colors = {'b', 'r', 'g'};
markers = {'o', 's', 'd'};
for btw = 1:length(numPam)
    semilogy(range_snr, ser_pam(btw, :), [colors{btw} '-' markers{btw}], 'DisplayName', sprintf('PAM symboER (M =
%d)', numPam(btw)));
    hold on;
    semilogy(range_snr, ber_pam(btw, :), [colors{btw} '--' markers{btw}], 'DisplayName', sprintf('PAM bitER (M =
%d)', numPam(btw)));
    semilogy(range_snr, ser_psk(btw, :), [colors{btw} ':' markers{btw}], 'DisplayName', sprintf('PSK symboER (M =
%d)', numPam(btw)));
    semilogy(range_snr, ber_psk(btw, :), [colors{btw} '-.' markers{btw}], 'DisplayName', sprintf('PSK bitER (M = %d)',
numPam(btw)));
end
xlabel('SNR per Bit');
ylabel('Probability of Error');
title('Comparison of PAM and PSK');
legend('Location', 'best');
grid on;
hold off;
```