POLITECNICO
MILANO 1863

IMPACT OF NON-GAUSSIAN NOISE ON LDPC PERFORMANCE

Stefano Biccari 10652373
Karsa Alizadeh 10874875
Pouria Saadatikhoshrou 10972532

INDEX

Introduction

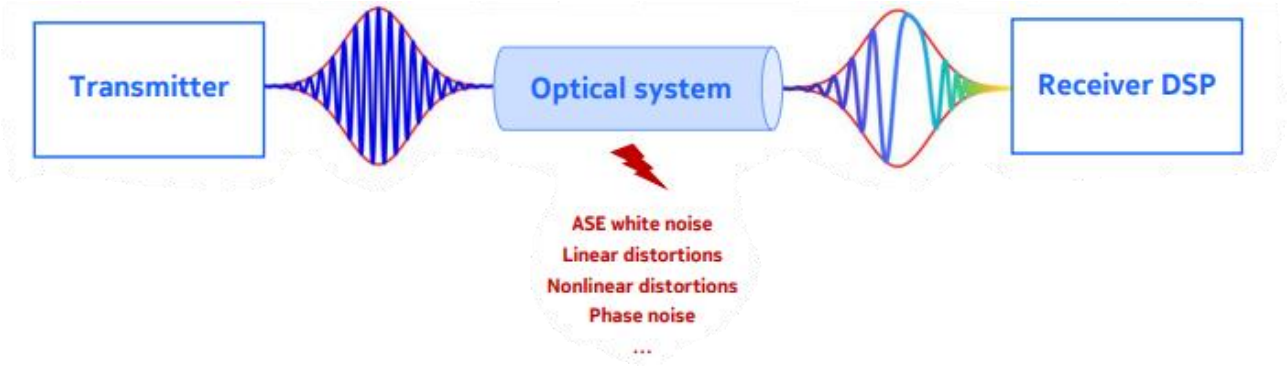
In digital communication, achieving high data transmission rates is extremely important, but this alone is not enough as we need the transmitted data to be received correctly. Hence, the interest in having reliable transmission over noisy channels.

But how can we achieve this? This is where LDPC codes, introduced by Robert G. Gallager, come into play. Gallager was an electrical engineer who discovered this code in the early 1960s, but due to its complexity, it was not used for the next 30 years. These codes have since become a cornerstone in error correction due to their near-capacity performance and efficient decoding algorithms.

When combined with high-order modulation schemes like 64-QAM, as in our case, LDPC codes can significantly enhance the throughput of communication systems. This study focuses on evaluating the BER and SNR performance of an LDPC-coded 64-QAM system under AWGN and non-Gaussian noise, modeled using the T-distribution.

What is the purpose of this study? When transmitting a signal along optical fiber, we encounter both linear and nonlinear impairments. For this reason, at the receiver, the signal is equalized using the DSP algorithm before decoding. After using the DSP algorithm, due to the aforementioned impairments, the resulting noise distribution affecting the symbols can no longer be considered strictly Gaussian, as it will exhibit heavier tails (the following are shown: a simplified diagram of the system). For this reason, our goal is to observe how the system changes from an ideal situation with AWGN (Additive White Gaussian Noise) to a situation where the noise dispersion is no longer Gaussian.

A simple representation of the of the situation is:

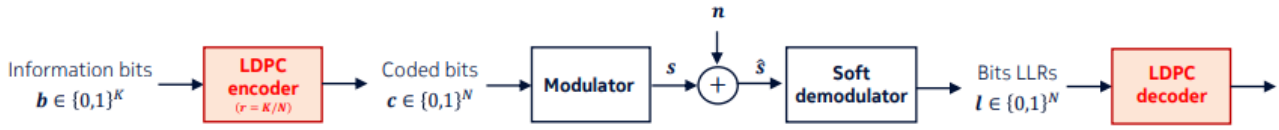


Objectives

The main objectives set before us are twofold:

1. Build a system using MATLAB to simulate the performance with AWGN (the parameters for that are given) and an LDPC encoding and decoding scheme. We need to use 64-QAM modulation.
2. Starting from the previously built system, consider the case where the noise distribution (the parameters for that are given) is no longer Gaussian and discuss how the performance is expected to change compared to the first case.

We can identify the system under examination as described below:



System Model and Theoretical Background

LDPC Codes:

LDPC (Low-Density Parity-Check) codes are error-correcting codes used in modern digital communications. They improve the reliability of data transmission even over noisy channels. Introduced by Robert G. Gallager in his doctoral thesis in the 1960s, they were not utilized for the next 30 years due to their complexity. In the 1990s, with the advent of more powerful computational capabilities, LDPC codes were exploited for their efficiency, as their performance approaches the theoretical limits of channel capacity described by Shannon's theorem.

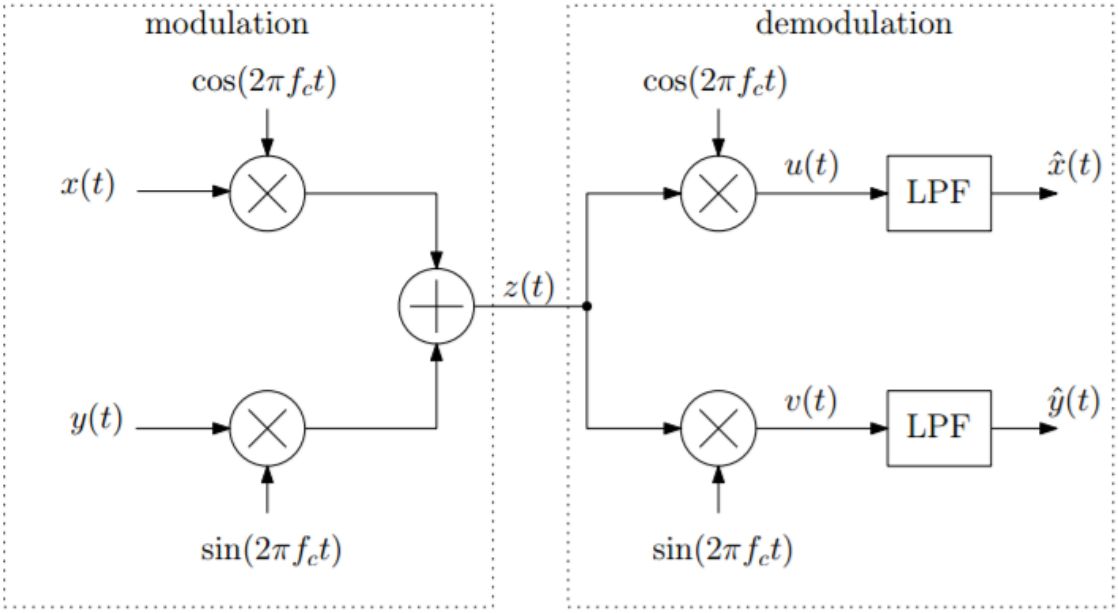
These codes are characterized by a sparse parity-check matrix, which enables efficient decoding. This decoding is performed using iterative algorithms, such as the Sum-Product Algorithm, which sums messages. LDPC decoding is computationally more efficient compared to convolutional error correction codes. While the encoding part involves the multiplication of information bits by a generating matrix derived from the parity check matrix.

Advantages include high error correction efficiency and the ability to approach channel capacity, though these benefits come at the cost of increased encoding complexity and higher computational power requirements for decoding.

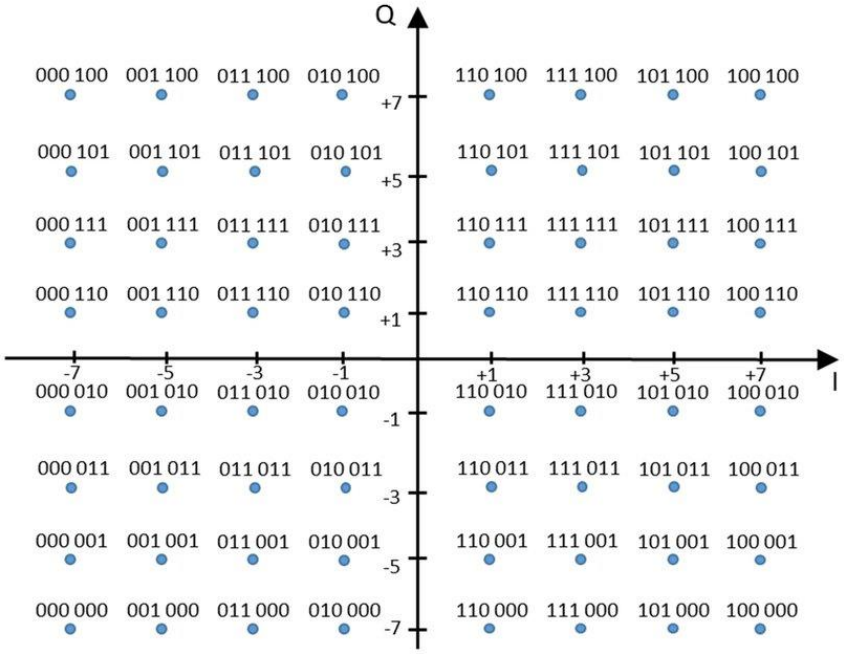
64-QAM Modulation:

64-QAM (64-Quadrature Amplitude Modulation) is a type of modulation used in both wireless and wired transmissions to increase the efficiency of data transmission.

Quadrature Amplitude Modulation refers to a modulation that is able to combine amplitude modulation and phase modulation during transmission without these interfering with each other due to the quadrature concept (thanks a modulation demodulation and filtering). This allows a higher number of bits per symbol to be transmitted compared to simple systems such as BPSK. A simple scheme is:



For 64 we speak instead of constellation, i.e. the number of possible symbol states. In our case, we will have a transmission of 6 bits/symbol. Where the points that you can see are the symbols and, as you can notice, all of them can be recognize with 6 bit:



This leads to greater spectral efficiency, allowing us to transmit more data per unit bandwidth, and to have a high transmission speed. It should be noted, however, that the larger the constellation, the more susceptible we are to noise that can lead us to have the wrong symbol, which is why 64-QAM modulation requires a higher SNR than 32-QAM, 16-QAM etc....

So what we can say is that, in our system, if the noise is too invasive, using a modulation with a lower constellation could result in an improvement in transmission as far as BER is concerned but a worsening as far as speed is concerned.

Noise Models:

Noise' refers to any unwanted interference or disturbance that affects signal transmission between sender and receiver. An unavoidable phenomenon caused by various factors such as thermal agitation present in electronic components such as amplifiers, electromagnetic interference given by other devices. More precisely, we can divide them into thermal noise, given by electronic devices that increase with increasing temperature; and quantum noise that is caused by the discrete nature of photons in optical transmissions.

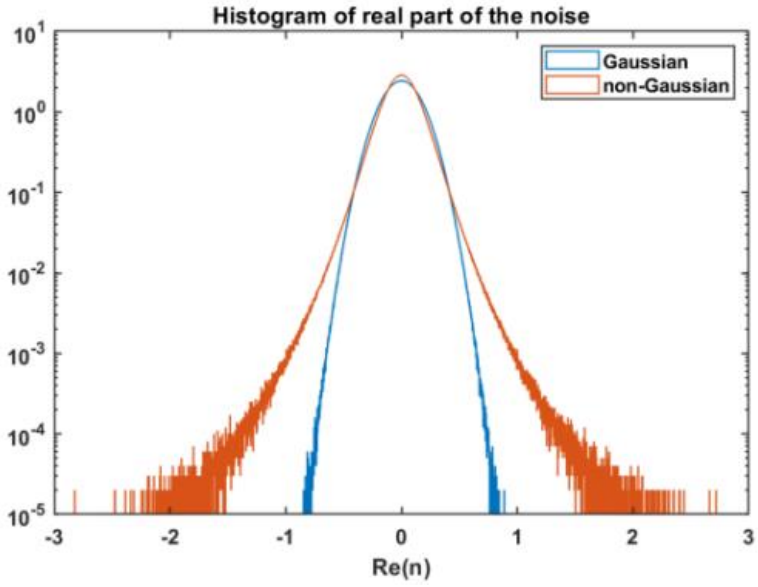
They are problematic as they increase the BER (Bit Error Rate) and decrease the SNR (Signal to Noise Ratio), thus limiting system performance.

In systems, it is necessary to mitigate the errors produced precisely because of noise, which in our case we do by using LDPC coding.

The noises we are considering in our case are:

- Additive White Gaussian Noise (AWGN): i.e. the ideal case (without considering a case where noise does not exist) as it has a Gaussian distribution and a uniform power density at all frequencies.
- non-Gaussian noise: this does not follow a normal distribution and can vary significantly in its statistical characteristics. In our case it is modelled using the T-distribution to represent impulsive noise environments. This noise has heavier tails compared to Gaussian noise, making it more challenging for communication systems.

We can see below a simple graphic representation of the two noises:



From Nokia presentation

Bit Error Rate (BER):

BER is the ratio of bit errors to the total number of transmitted bits. It serves as a critical performance metric for evaluating the effectiveness of error correction codes and modulation schemes.

Signal to Noise Ratio (SNR)

A measure used in telecommunications and electronics to assess the quality of a signal in relation to the noise level present in the environment or transmission system.

To do so, the SNR is obtained from the ratio of the useful signal power divided by the noise power the signal has gained during transmission.

This number is generally expressed in dB.

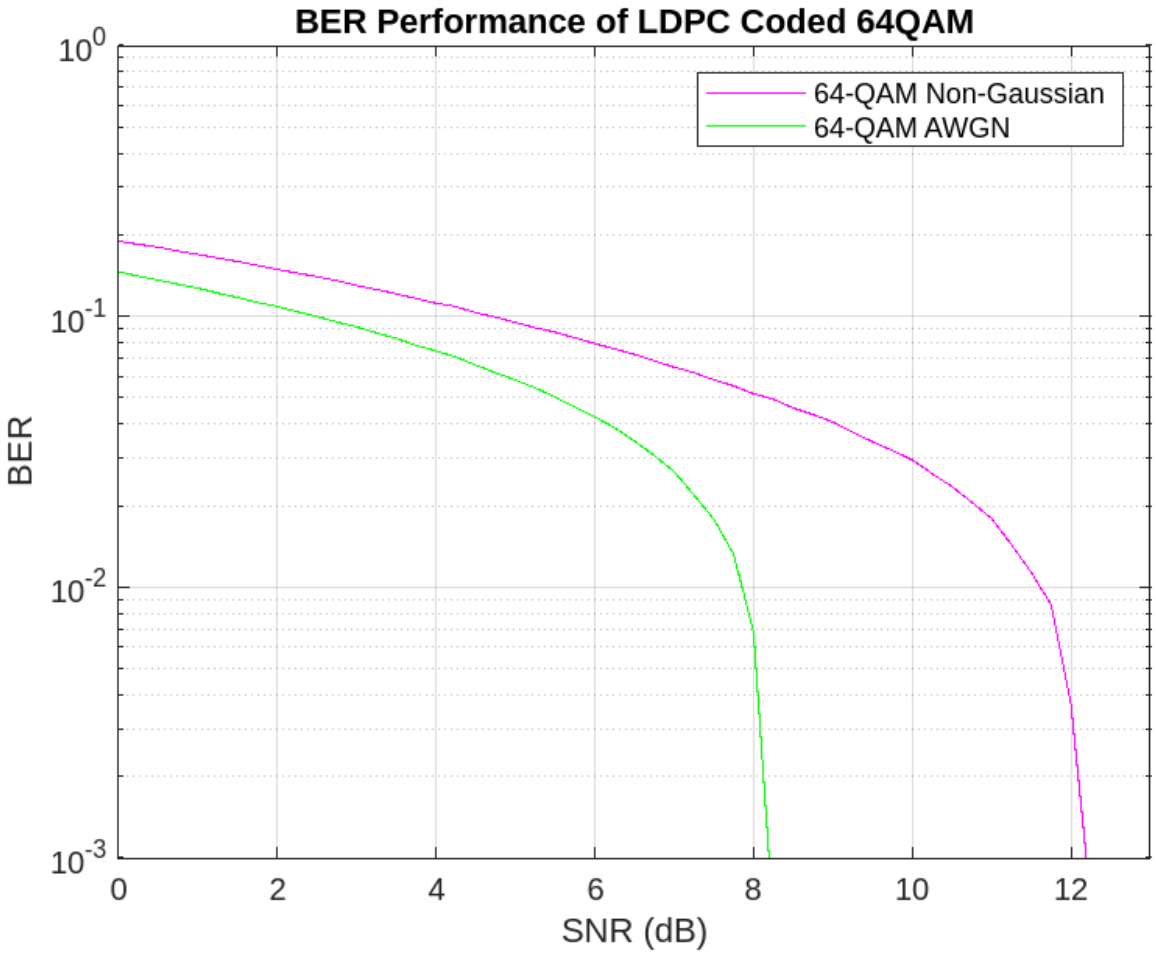
Digital Signal Processing (DSP):

DSP algorithm (Digital Signal Processing) are various techniques and algorithms used to manipulate digital signals with the aim of improving their quality, efficiency or interpretation. In our case we are mainly interested in that it performs 4 operations:

- Filtering: used to remove unwanted noise or unwanted frequency components from the signal.
- Transform: Fourier Transform and Laplace Transform used to represent signals in the frequency domain, facilitating analysis and filtering.
- Equalization: Used to compensate for signal distortions caused by the transmission channel, thus improving the quality of the received signal (the one of main interest to us here as it allows us to mitigate impairments in the received signal)
- Compression: Data compression techniques to reduce the size of data without significant loss of information, used for example in audio and video compression.

Results and Discussion

The BER performance was evaluated for both AWGN and non-Gaussian noise conditions across a range of SNRs. The simulation results are plotted on a semi-logarithmic scale, providing insights into the system's robustness and error correction capabilities. We can see below a simple graphic representation of the result:



Findings:

1. AWGN Performance: The BER decreases significantly with increasing SNR, showcasing the effectiveness of LDPC codes in mitigating Gaussian noise.
2. Non-Gaussian Performance: The system experiences higher BER under non-Gaussian noise at the same SNR levels, highlighting the challenges posed by impulsive noise.

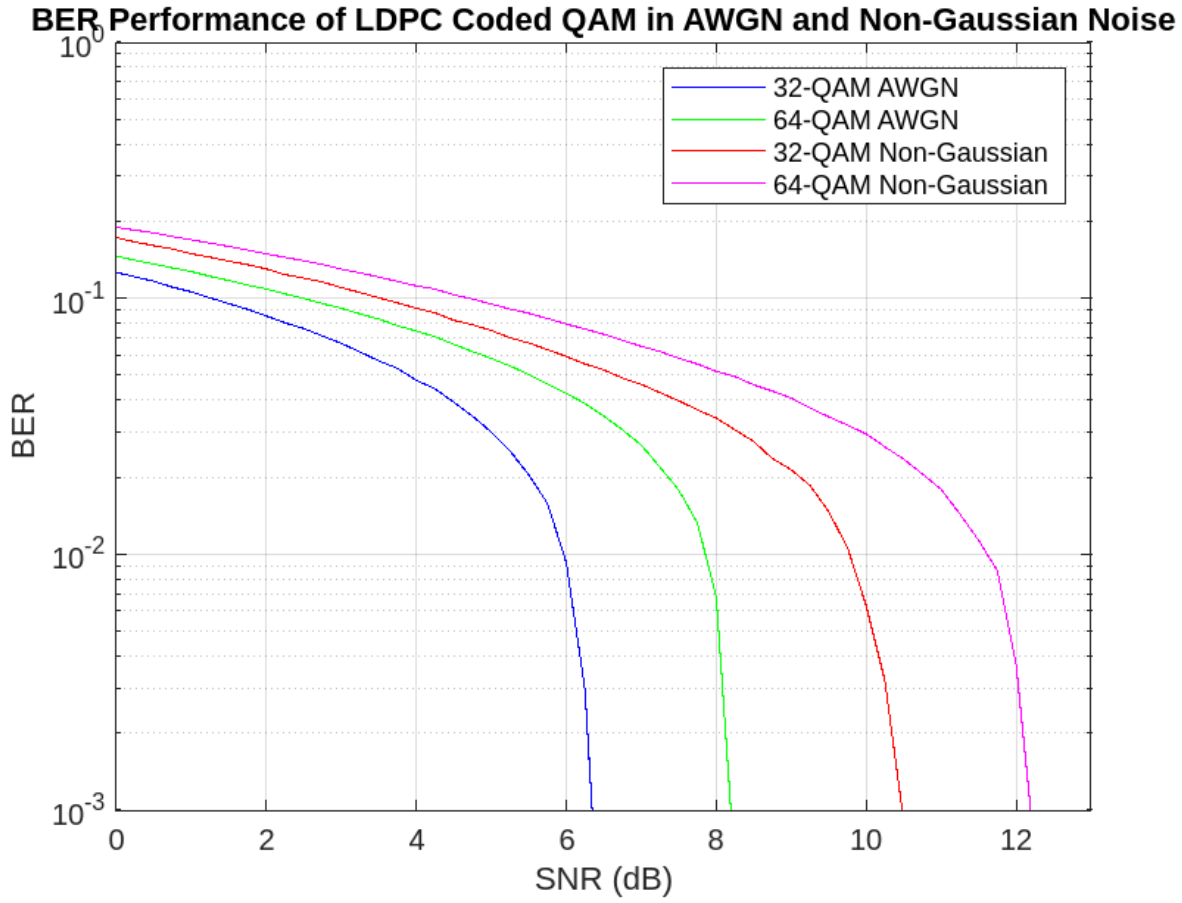
Plot Interpretation:

- The plots indicate that while LDPC codes improve performance under both noise conditions, non-Gaussian noise requires higher SNR to achieve comparable BER to AWGN.

Project modification

Although not required by the original design, we wondered how the situation would actually change if, with the current parameters, we used a 32 QAM instead of a 64QAM.

The graphs obtained from this test are as follows:



As one might expect from the brief explanation made in retrospect, what we get is a situation in which BER will be lower for the same useful signal power emitted. This is precisely because the constellation allows us to have a lower error rate.

The choice, however, to use the 32 QAM will result in a reduction in performance with respect to the bit rate as we go from transmitting 6 bits/symbol to 5 bits/symbol.

In the case of non-Gaussian noise, however, it is interesting to see that the BER decreases much faster between the 64QAM and 32QAM cases. This is precisely because non-Gaussian noise is difficult to handle as it is practically unpredictable. This results in the 64QAM having significantly more errors for the same power output.

In order to best show the effect achieved by LDPC coding, we thought we would also study the situation in which it is not used. We can notice that:

1. **Problem for the error Correction:** The absence of LDPC encoding and decoding means there is no error correction capability. Each bit error introduced by noise will directly impact the BER.

2. **BER Performance:**

    - **Higher BER:** Without LDPC coding, the BER will be significantly higher, especially at lower Eb/N0 values. This is because LDPC codes provide robust error correction, which can correct many of the bit errors caused by noise.

    - **No Error Correction:** The modulation and noise addition steps are the same, but now we rely solely on the inherent robustness of the modulation scheme against noise, without any additional error correction.

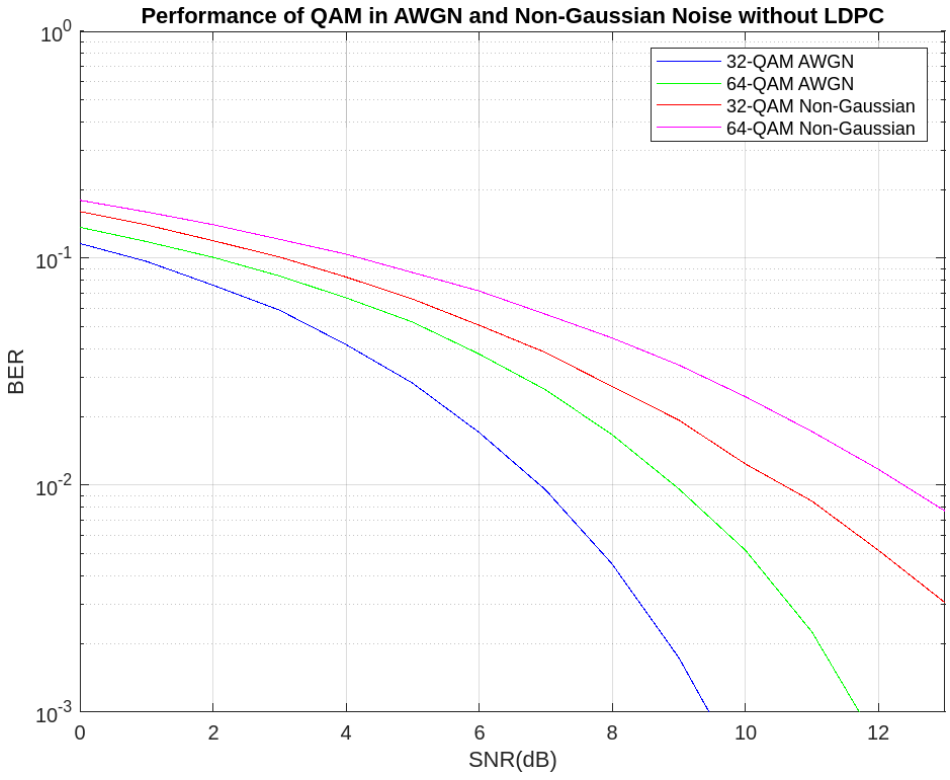
    - **Modulation Order Impact:** Higher order modulations (like 64-QAM) will have a worse BER compared to lower order modulations (like 32-QAM) at the same Eb/N0, as they are more susceptible to noise.

3. **Results Comparison:**

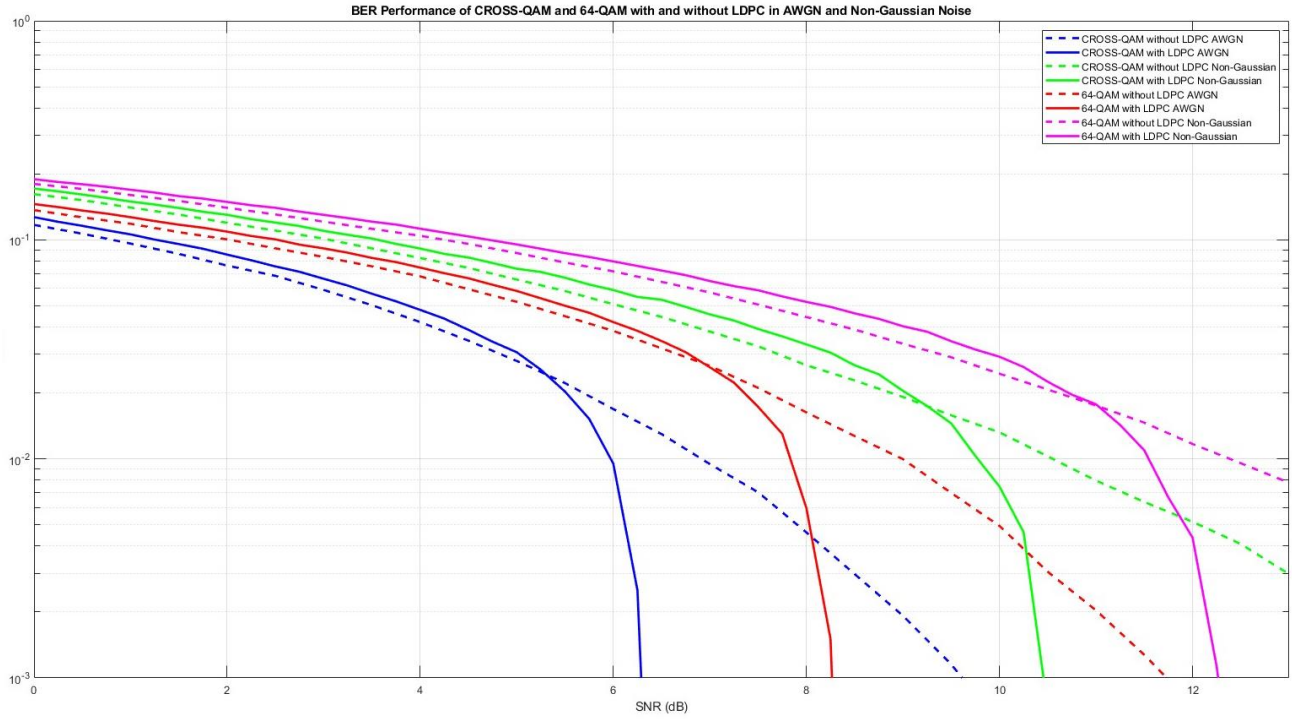
    - **AWGN Channel:** BER in AWGN channels will be higher than with LDPC due to the lack of error correction.

Non-Gaussian Noise: The impact of non-Gaussian noise will be even more pronounced, as the noise has a heavier tail distribution (more extreme values), leading to higher BER without error correction.

In our system we have that result:



Tha compared to the previous graph we obtain:



As you can see, if we want to have a BER like 0,01 in the case of a code that uses LDPC we need to have less SNR. That shows how LDPC can improve the situation.

Implementation

MATLAB Code:

```
%% Parameters for uncoded
K = 58320; % Number of information bits
numFrames = 100; % Number of frames for simulation
EbN0_dB_uncoded = 0:0.5:13; % Eb/N0 range in dB
modulationSchemes = [32, 64]; % Modulation orders
%% Parameters for LDPC coded
K_ldpc = 58320; % Number of information bits for DVB-S2 LDPC
R = 9/10; % Code rate
N = K_ldpc / R; % Number of coded bits
EbN0_dB_coded = 0:0.25:13; % Eb/N0 range in dB
maxNumIter = 100; % Increased number of iterations for LDPC decoding
%% LDPC Encoder and Decoder Configuration
ldpcEncoderCfg = ldpcEncoderConfig(dvbs2ldpc(R)); % Create LDPC encoder
configuration object
ldpcDecoderCfg = ldpcDecoderConfig(dvbs2ldpc(R)); % Create LDPC decoder
configuration object
%% BER Calculation
berAWGN_uncoded = zeros(length(modulationSchemes), length(EbN0_dB_uncoded));
berNonGaussian_uncoded = zeros(length(modulationSchemes),
length(EbN0_dB_uncoded));
berAWGN_coded = zeros(length(modulationSchemes), length(EbN0_dB_coded));
berNonGaussian_coded = zeros(length(modulationSchemes), length(EbN0_dB_coded));
%% Progress bar initialization
totalIterations = 2 * (length(modulationSchemes) * length(EbN0_dB_uncoded) *
numFrames + length(modulationSchemes) * length(EbN0_dB_coded) * numFrames);
currentIteration = 0;
h = waitbar(0, 'Initializing simulation...');
%% Simulation for uncoded QAM
for modIdx = 1:length(modulationSchemes)
   M = modulationSchemes(modIdx);
   bitsPerSymbol = log2(M);
   for i = 1:length(EbN0_dB_uncoded)
       EbN0 = 10^(EbN0_dB_uncoded(i)/10);
       noiseVar = 1/(2*EbN0*bitsPerSymbol); % Adjust noise variance based on
bits per symbol
       for frame = 1:numFrames
           currentIteration = currentIteration + 1;
           progress = currentIteration / totalIterations;
           waitbar(progress, h, sprintf('Simulating: %.2f%% complete', progress
* 100));

           % Generate random information bits with the required size
           b = randi([0 1], K, 1);

           % QAM Modulation
           modData = qammod(b, M, 'InputType', 'bit', 'UnitAveragePower', true);

           % AWGN Noise
```

```
            noiseAWGN = sqrt(noiseVar/2) * (randn(size(modData)) +
1i*randn(size(modData)));

            % Non-Gaussian Noise (T-Distribution)
            nu = 6; % Degrees of freedom
            noiseNonGaussian = sqrt(noiseVar * (nu-2)/nu) * (trnd(nu,
size(modData)) + 1i*trnd(nu, size(modData)));

            % Received signal
            rxSigAWGN = modData + noiseAWGN;
            rxSigNonGaussian = modData + noiseNonGaussian;

            % Soft Demodulation
            llrAWGN = qamdemod(rxSigAWGN, M, 'OutputType', 'approxllr',
'UnitAveragePower', true, 'NoiseVariance', noiseVar);
            llrNonGaussian = qamdemod(rxSigNonGaussian, M, 'OutputType',
'approxllr', 'UnitAveragePower', true, 'NoiseVariance', noiseVar);

            % BER Calculation
            berAWGN_uncoded(modIdx, i) = berAWGN_uncoded(modIdx, i) + sum(b ~=
double(llrAWGN < 0))/K;
            berNonGaussian_uncoded(modIdx, i) = berNonGaussian_uncoded(modIdx, i)
+ sum(b ~= double(llrNonGaussian < 0))/K;
        end
        berAWGN_uncoded(modIdx, i) = berAWGN_uncoded(modIdx, i) / numFrames;
        berNonGaussian_uncoded(modIdx, i) = berNonGaussian_uncoded(modIdx, i) /
numFrames;
    end
end
%% Simulation for LDPC coded QAM
for modIdx = 1:length(modulationSchemes)
    M = modulationSchemes(modIdx);
    bitsPerSymbol = log2(M);
    for i = 1:length(EbN0_dB_coded)
        EbN0 = 10^(EbN0_dB_coded(i)/10);
        noiseVar = 1/(2*R*bitsPerSymbol*EbN0);
        errBitsAWGN = 0; % Initialization of error counting for AWGN
        errBitsNonGaussian = 0; % Initialization of error counting for Non-
Gaussian
        for frame = 1:numFrames
            currentIteration = currentIteration + 1;
            progress = currentIteration / totalIterations;
            waitbar(progress, h, sprintf('Simulating: %.2f%% complete', progress
* 100));

            % Generate random information bits with the required size
            b = randi([0 1], K_ldpc, 1); % Generate a vector of size K_ldpc x 1

            % LDPC Encoding
            c = ldpcEncode(b, ldpcEncoderCfg);

            % QAM Modulation
            modData = qammod(c, M, 'InputType', 'bit', 'UnitAveragePower', true);
```

```
            % AWGN Noise
            noiseAWGN = sqrt(noiseVar/2) * (randn(size(modData)) +
1i*randn(size(modData)));

            % Non-Gaussian Noise (T-Distribution)
            nu = 6; % Degrees of freedom
            noiseNonGaussian = sqrt(noiseVar * (nu-2)/nu) * (trnd(nu,
size(modData)) + 1i*trnd(nu, size(modData)));

            % Received signal
            rxSigAWGN = modData + noiseAWGN;
            rxSigNonGaussian = modData + noiseNonGaussian;

            % Soft Demodulation
            llrAWGN = qamdemod(rxSigAWGN, M, 'OutputType', 'approxllr',
'UnitAveragePower', true, 'NoiseVariance', noiseVar);
            llrNonGaussian = qamdemod(rxSigNonGaussian, M, 'OutputType',
'approxllr', 'UnitAveragePower', true, 'NoiseVariance', noiseVar);

            % LDPC Decoding
            b_hat_AWGN = ldpcDecode(llrAWGN, ldpcDecoderCfg, maxNumIter);
            b_hat_NonGaussian = ldpcDecode(llrNonGaussian, ldpcDecoderCfg,
maxNumIter);

            % BER Calculation
            errBitsAWGN = errBitsAWGN + sum(b ~= b_hat_AWGN);
            errBitsNonGaussian = errBitsNonGaussian + sum(b ~=
b_hat_NonGaussian);
        end
        berAWGN_coded(modIdx, i) = errBitsAWGN / (K_ldpc * numFrames);
        berNonGaussian_coded(modIdx, i) = errBitsNonGaussian / (K_ldpc *
numFrames);
    end
end
% Close the progress bar
close(h);
%% Plotting Results
% 1. Figure with two plots for 64-QAM (LDPC coded and uncoded)
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(2, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berAWGN_coded(2, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(2, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('64-QAM AWGN without LDPC', '64-QAM AWGN with LDPC', '64-QAM Non-Gaussian
without LDPC', '64-QAM Non-Gaussian with LDPC');
title('BER Performance of 64-QAM (LDPC Coded vs Uncoded) in AWGN and Non-
Gaussian Noise');
xlim([0 13]);
```

```
ylim([1e-3 1]);
hold off;
% 2. Figure with two plots for 32-QAM (LDPC coded and uncoded)
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berAWGN_coded(1, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(1, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(1, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM AWGN without LDPC', '32-cross QAM AWGN with LDPC', '32-
cross QAM Non-Gaussian without LDPC', '32-cross QAM Non-Gaussian with LDPC');
title('BER Performance of 32-QAM (LDPC Coded vs Uncoded) in AWGN and Non-
Gaussian Noise');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 3. Figure with one plot for 64-QAM and 32-QAM in LDPC coded
figure;
semilogy(EbN0_dB_coded, berAWGN_coded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berAWGN_coded(2, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(1, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM LDPC coded AWGN', '64-QAM LDPC coded AWGN', '32-cross QAM
LDPC coded Non-Gaussian', '64-QAM LDPC coded Non-Gaussian');
title('BER Performance of LDPC Coded QAM in AWGN and Non-Gaussian Noise');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 4. Figure with one plot for 64-QAM and 32-QAM without LDPC
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(2, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(1, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM without LDPC AWGN', '64-QAM without LDPC AWGN', '32-cross
QAM without LDPC Non-Gaussian', '64-QAM without LDPC Non-Gaussian');
title('BER Performance of Uncoded QAM in AWGN and Non-Gaussian Noise');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 5. Figure with one plot for 64-QAM in LDPC coded
figure;
```

```
semilogy(EbN0_dB_coded, berAWGN_coded(2, :), 'r', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berNonGaussian_coded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('64-QAM LDPC coded AWGN', '64-QAM LDPC coded Non-Gaussian');
title('BER Performance of 64-QAM in AWGN and Non-Gaussian Noise (LDPC coded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 6. Figure with one plot for 64-QAM in without LDPC coded
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(2, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(2, :), 'g', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('64-QAM without LDPC AWGN', '64-QAM without LDPC Non-Gaussian');
title('BER Performance of 64-QAM in AWGN and Non-Gaussian Noise (uncoded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 7. Figure with one plot for 32-cross QAM in LDPC coded
figure;
semilogy(EbN0_dB_coded, berAWGN_coded(1, :), 'r', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berNonGaussian_coded(1, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM LDPC coded AWGN', '32-cross QAM LDPC coded Non-Gaussian');
title('BER Performance of 32-cross QAM in AWGN and Non-Gaussian Noise (LDPC
coded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 8. Figure with one plot for 32-cross QAM in without LDPC coded
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(1, :), 'g', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM without LDPC AWGN', '32-cross QAM without LDPC Non-
Gaussian');
title('BER Performance of 32-cross QAM in AWGN and Non-Gaussian Noise
(uncoded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
```

```
% 9. Figure with one plot for 32-cross QAM and 64-QAM in without LDPC coded
figure;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_uncoded, berAWGN_uncoded(2, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(1, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_uncoded, berNonGaussian_uncoded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM without LDPC AWGN', '64-QAM without LDPC AWGN', '32-cross
QAM without LDPC Non-Gaussian', '64-QAM without LDPC Non-Gaussian');
title('BER Performance of 32-cross QAM and 64-QAM in AWGN and Non-Gaussian Noise
(uncoded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
% 10. Figure with one plot for 32-cross QAM and 64-QAM in with LDPC coded
figure;
semilogy(EbN0_dB_coded, berAWGN_coded(1, :), 'b', 'LineWidth', 2);
hold on;
semilogy(EbN0_dB_coded, berAWGN_coded(2, :), 'r', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(1, :), 'g', 'LineWidth', 2);
semilogy(EbN0_dB_coded, berNonGaussian_coded(2, :), 'm', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('BER');
legend('32-cross QAM LDPC coded AWGN', '64-QAM LDPC coded AWGN', '32-cross QAM
LDPC coded Non-Gaussian', '64-QAM LDPC coded Non-Gaussian');
title('BER Performance of 32-cross QAM and 64-QAM in AWGN and Non-Gaussian Noise
(LDPC coded)');
xlim([0 13]);
ylim([1e-3 1]);
hold off;
```

Conclusion

This study demonstrates the implementation and analysis of a 64-QAM/32-QAM communication system with LDPC coding in the presence of AWGN and non-Gaussian noise. The results highlight the robustness of LDPC codes and the impact of different noise patterns on system performance.

The most obvious conclusion is that, to have a given BER when having a channel with non-Gaussian noise distribution, we will also need to have a higher SNR, which translates into a higher.

The future work could explore adaptive modulation and coding schemes to further improve the reliability of communication in various noise environments.

(here you can find our material)