EC2: Communication Theory - Course Project

Autrio Das
2022112007
International Institute of Information Technology Hyderabad
autrio.das@research.iiit.ac.in

Pearl Shah
2022102073
International Institute of Information Technology Hyderabad
pearl.shah@students.iiit.ac.in

***Abstract*—This project explores the simulation of a comprehensive communication model implemented in MATLAB, designed to evaluate different digital modulation techniques. The core objective is to convert a provided audio file into a bitstream using an Analog-to-Digital (A/D) converter, then encode these bits into symbols through various encoding schemes based on the team number modulo operation. Notable encoding techniques include 4-ary Amplitude-Shift Keying (ASK), Frequency-Shift Keying (FSK), Binary Phase-Shift Keying (BPSK), two types of Quadrature Phase-Shift Keying (QPSK), and 16-ary Quadrature Amplitude Modulation (16-QAM) from which we are required to do ASK. Subsequent stages involve line coding, modulation, and transmission through two types of Additive White Gaussian Noise (AWGN) channels—one memoryless and the other with memory. Demodulation, decoding, and a detailed analysis of the output signals, including Signal-to-Noise Ratio (SNR), Bit Error Rate (BER), and power efficiency, are conducted to gauge the performance of the communication system under various noise conditions. This project leverages MATLAB's robust(?) simulation capabilities to visually and quantitatively analyze the impact of different encoding and modulation strategies on communication system performance, providing essential insights into the practical applications and challenges in digital communications.**

I. Introduction

In the rapidly evolving field of digital communications, simulating realistic communication systems provides invaluable insights into the practical applications and theoretical underpinnings of modern communication technologies. This project, conducted as a part of the Communication Theory course, Spring 2024, focuses on the implementation and analysis of a digital communication model using MATLAB. Specifically, our project is centered around the use of 4-ary Amplitude-Shift Keying (ASK), a modulation technique where digital data is represented as variations in the amplitude of a carrier wave.

The primary objective of this project is to recreate a specific audio file from its transmission through a simulated digital communication system. The process begins with the audio file being converted into a digital format via an Analog-to-Digital (A/D) converter. This digital data is then encoded using the ASK scheme, processed through various stages of the communication model, and finally reconstructed back into an audio output. The ability to accurately recreate the original audio file after it has been subjected to various transformations within the model serves as a testament to the effectiveness and accuracy of the simulation.

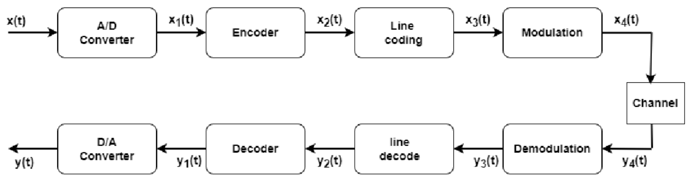

Fig. 1. System Model

This project not only enhances our understanding of ASK but also allows us to explore the effects of different system components on the fidelity of the transmitted audio. Through this exercise, we aim to bridge the gap between theoretical knowledge and practical application, providing a hands-on experience in the simulation of digital communication systems.

II. A/D converter

The A/D converter block serves as the initial stage in the communication model, tasked with converting the provided audio file, denoted as x, into a digital format suitable for further processing. The input to this block,x, represents the analog signal extracted from the audio file. This analog signal is then sampled at regular intervals to produce discrete amplitude values, effectively quantizing the continuous-time waveform into a discrete-time representation.

The conversion process entails two primary steps: sampling and quantization. During sampling, the continuous-time signal is discretized in time, capturing its amplitude at specific instances determined by the sampling rate. The resulting sequence of samples represents a digital approximation of the original analog waveform.

Following sampling, quantization is performed to represent each sample with a finite number of bits. This step involves dividing the continuous range of sample amplitudes into discrete levels and assigning a digital code to each level. The number of bits used for quantization determines the resolution of the digital representation, with higher bit depths offering finer amplitude granularity but requiring increased computational resources.

We use inbuilt MATLAB function to implement A/D converter. Upon completion of sampling and quantization, the A/D converter block generates the digital output, denoted as x1, which comprises a sequence of binary digits (bits) representing the discrete amplitude values of the input signal x. This digital representation enables subsequent processing stages within the communication model to manipulate and transmit the audio signal in a digital domain. The A/D converter is implemented as such:

Listing 1. A_D_Converter.m

```
function [x1,Fs] = A_D_converter(filename)

  [audio,f] = audioread(filename);
  binAudio = dec2bin( typecast(
      single(audio(:)), 'uint8'), 8 ) - '0';
  lenB = length(binAudio);
  x1 = reshape(binAudio,lenB*8,1);
  Fs = f;
  x1 = x1(1:72000);

end
```

III. Encoder

The Encoder block, operating under the paradigm of 4-ary amplitude-shift keying (ASK), is responsible for transforming the incoming bit stream or sequence into a sequence of symbols denoted as a_k. This modulation technique facilitates the mapping of pairs of consecutive bits to specific amplitude levels, thereby encoding digital information into analog waveforms for transmission over a communication channel.

The encoding process adheres to a predefined mapping scheme, where each pair of consecutive bits corresponds to a distinct amplitude level. Specifically, the following mapping scheme is employed: Accordingly, the Encoder block interprets every two consecutive bits from the input bit stream and assigns them a corresponding amplitude level based on the aforementioned mapping scheme. For instance, the bit sequence ”00” is mapped to an amplitude level of 0, ”01” to 1, ”10” to 2, and ”11” to 3.

$$00 \rightarrow 0$$
$$01 \rightarrow 1$$
$$10 \rightarrow 2$$
$$11 \rightarrow 3$$

This encoding process effectively translates digital information into analog symbols represented by different amplitude levels. The resulting sequence of symbols, denoted as a_k , embodies the modulated signal ready for transmission through the communication channel.

In summary, the Encoder block implements 4-ary amplitude-shift keying (ASK) modulation by mapping pairs of consecutive bits to specific amplitude levels, thereby encoding digital data into analog symbols for transmission.

Listing 2. encoder.m

```
function map = encoder(x1)

bitstream = x1;

if mod(numel(bitstream), 2) ~= 0
   disp('Warning: Number of elements is not
       even. Last element will be ignored.');
   bitstream = bitstream(1:end-1); % Remove
       the last element
end

bits = reshape(bitstream, 2, []);

map(all(bits==[0;0])) = 0;
map(all(bits==[0;1])) = 1;
map(all(bits==[1;0])) = 2;
map(all(bits==[1;1])) = 3;

map = map';
end
```

A. Ideal BER for given SNR

Finding the relation between BER and SNR involves the following derivation: In the case of 4-Ary Amplitude Shift

Keying (ASK) the message are transmitted by the pulses

$$0, p(t), 2p(t) \dots (M-1)p(t)$$
$$\text{in our case} \implies 0, p(t), 2p(t), 3p(t)$$

if E_p is the energy of one pulse and all the pulses are equally likely, average pulse energy E_{pM} is given by

$$E_{pM} = \frac{1}{M}\left[E_p + 4E_p + 9E_p\right]$$

Which approximately equates to

$$E_{pM} \approx \frac{M^2-1}{6}E_p \implies \frac{5}{2}E_p$$

The bit energy is

$$E_b = \frac{E_p M}{\log_2 M} = \frac{5}{2\log_2 M}E_p$$

To calculate P_{eM} , we observe that the case of the two extreme symbols [represented by 0, (M - 1)p(t)] is similar to the binary case because they have to guard against only one neighbor. As for the remaining symbols, they must guard against neighbors on both sides, and, hence, $P(\epsilon|m)$ in this case is twice that of the extreme symbol. From Fig. 10.23a it is evident that $P(\epsilon|m)$ is $Q(A_p|\sigma_n)$ for the two extreme signals and is $2Q(A_p|\sigma_n)$ for the remaining (M-2) symbols.

Hence,

$$p_{eM} = \sum_{i=1}^{M} P(m_i)P(\epsilon|m_i)$$
$$\implies \frac{1}{M}\sum_{i=1}^{M} P(\epsilon|m_i)$$
$$\implies \frac{2(M-1)}{M}Q(\frac{A_p}{\sigma_n})$$

If the reciever filter is a matched filter, we get

$$P_{eM} = \frac{2(M-1)}{M}Q\left[\sqrt{\frac{4\log_2 M}{5(M^2-1)}(\frac{E_b}{\mathcal{N}_0})}\right]$$

IV. Line Coding Scheme

Line coding is a crucial part of digital communication systems, transforming a sequence of bits into a digital signal suitable for transmission over a physical medium. The choice of line coding technique can significantly affect the performance and reliability of the communication system. In this project, we explore two different line coding schemes: the Rectangular Pulse and the Raised Cosine Pulse.

A. Rectangular Pulse

The simplest form of line coding implemented in this project is using a rectangular pulse. This method involves mapping each bit directly to a pulse of fixed duration and amplitude. The mathematical representation of the line coded signal using a rectangular pulse can be expressed as follows:

$$x_3(t) = \sum_{k=-\infty}^{\infty} a_k p_{\text{rect}}(t - kT_b)$$

where a_k denotes the symbol corresponding to the k^{th} bit, T_b is the bit duration, and $p_{\text{rect}}(t)$ is defined as:

$$p_{\text{rect}}(t) = \begin{cases} 1 & \text{if } 0 \le t < T_b \\ 0 & \text{otherwise} \end{cases}$$

This coding scheme is simple and easy to implement but is susceptible to issues such as baseline wander and lack of synchronization markers, which might affect the signal integrity in noisy channels.

B. Raised Cosine Pulse

To mitigate some of the issues faced with rectangular pulses, a Raised Cosine pulse is also employed. This pulse shape helps in reducing the bandwidth of the signal while minimizing intersymbol interference (ISI), a common problem in digital communications that occurs when pulses ”bleed” into one another. The Raised Cosine pulse is defined as:

$$p_{\text{rc}}(t) = \frac{\cos\left(\frac{\pi t}{T_b}\right)}{1 - \left(\frac{2t}{T_b}\right)^2}$$

where T_b is the symbol duration. The line coded signal using the Raised Cosine pulse is given by:

$$x_3(t) = \sum_{k=-\infty}^{\infty} a_k p_{\text{rc}}(t - kT_b)$$

The use of the Raised Cosine pulse greatly enhances the performance of the communication system by ensuring that the spectral components of the signal are confined within the desired frequency band, thus reducing the likelihood of channel-induced distortions.

C. PSD of Line Coded Signals

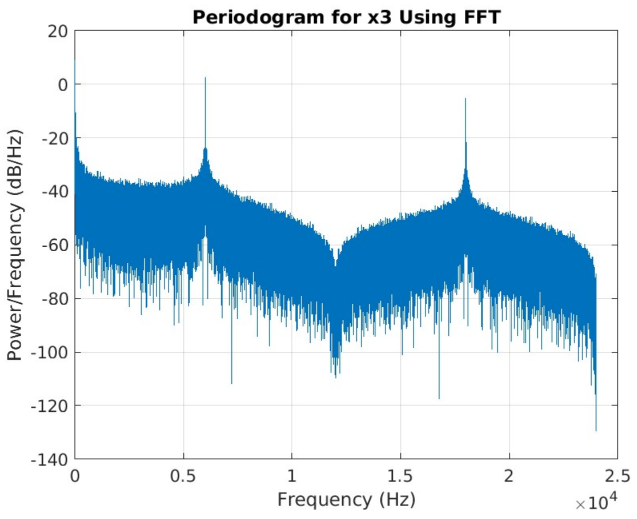

Fig. 2. PSD for line coded signal with rect function

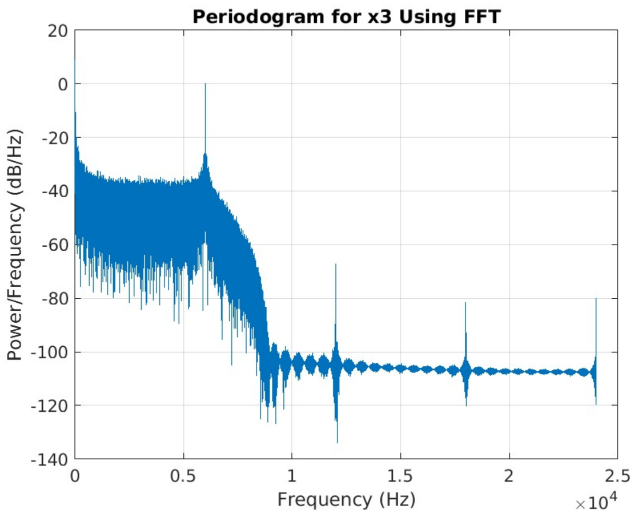

Fig. 3. PSD for line coded signal with raised cosine function

Listing 3. LineEncoder.m

```
function [x3,ht] = lineEncoder(x2,Tb,type)

oversampling_factor = 4;
m = oversampling_factor;
a = 0.5;
lenRC = 10;
[raised_cosine_filter, dummy] =
    raised_cosine(a, m, lenRC);
rect_filter = ones(1, m);

x2Ups = upsample(x2,m);

x3Cos =
    conv(x2Ups,raised_cosine_filter,"same");
```

```
x3Rect = conv(x2Ups,rect_filter,'same');

if(type=="raised cosine")
   x3 = x3Cos;
   ht = raised_cosine_filter;
elseif(type=="rect")
   x3= x3Rect;
   ht = rect_filter;
end

end
```

In conclusion, the choice of line coding scheme plays a pivotal role in defining the efficiency and robustness of the digital communication system. Through the simulations carried out in this project, the impacts of different pulse shapes on the signal integrity and system performance were thoroughly analyzed.

V. Modulation Scheme

Modulation is a fundamental step in any communication system, used to adapt the baseband signal for transmission over a physical medium. In this project, Amplitude-Shift Keying (ASK), a form of amplitude modulation, is employed. ASK is a robust modulation scheme where the amplitude of the carrier signal is varied in accordance with the input signal while keeping the frequency and phase constant.

A. Amplitude Modulation Techniques

The two amplitude modulation techniques considered for implementing ASK in this project are Double-Sideband Suppressed Carrier (DSB-SC) and Single-Sideband (SSB) modulation. Each technique offers distinct advantages and challenges:

1) Double-Sideband Suppressed Carrier (DSB-SC): In DSB-SC modulation, both sidebands are transmitted, but the carrier is suppressed, reducing the power usage as the carrier typically does not convey information. The mathematical representation of the DSB-SC modulated signal is:

$$s(t) = x(t) \cdot \cos(2\pi f_c t)$$

where $x(t)$ is the message signal, and f_c is the carrier frequency. The primary advantage of DSB-SC is the reduction in power consumption compared to traditional AM. However, the receiver design needs to be more complex as it must reconstruct the carrier to demodulate the signal.

2) Single-Sideband (SSB) Modulation: SSB modulation transmits only one of the sidebands (upper or lower), which further reduces the bandwidth and power requirements compared to DSB-SC. This is particularly beneficial in bandwidth-constrained systems. The SSB modulated signal can be expressed as: While SSB is efficient in terms of bandwidth and power, it requires even more precise filter and carrier synchronization at the receiver.

B. Choice of Modulation Technique

For this project, after careful consideration of the advantages and disadvantages, DSB-SC was chosen. This decision was made due to its relative simplicity compared to SSB and its adequacy for demonstrating basic ASK modulation principles in an educational setting. Furthermore, the suppression of the carrier in DSB-SC provides a clear demonstration of energy efficiency in modulation.

C. Simulation Results

Plots demonstrating the modulation process are crucial for visualizing the effects of the chosen modulation technique. In the report, plots of the signal after modulation using a carrier frequency of 100 Hz are included for visualization purposes. These plots help in understanding how the modulated signal behaves in the presence of a carrier and assists in analyzing the spectral characteristics of the modulated signal. For the final implementation of the communication model, a carrier frequency of 1 MHz will be used, adhering to typical communication standards. This higher frequency ensures the signal's compatibility with real-world communication channels.

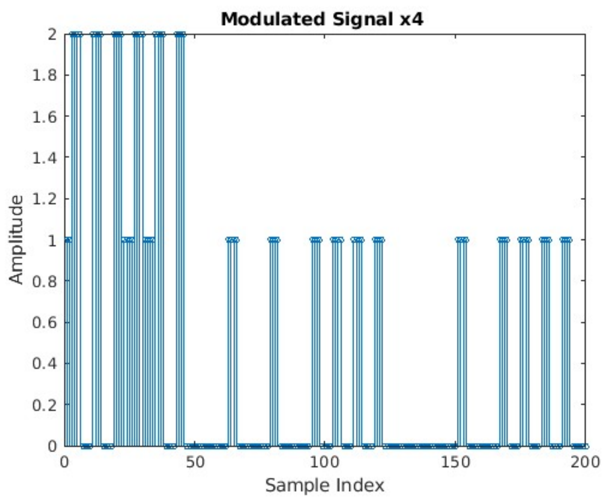

Fig. 4. 100 samples of modulated signal with rect function

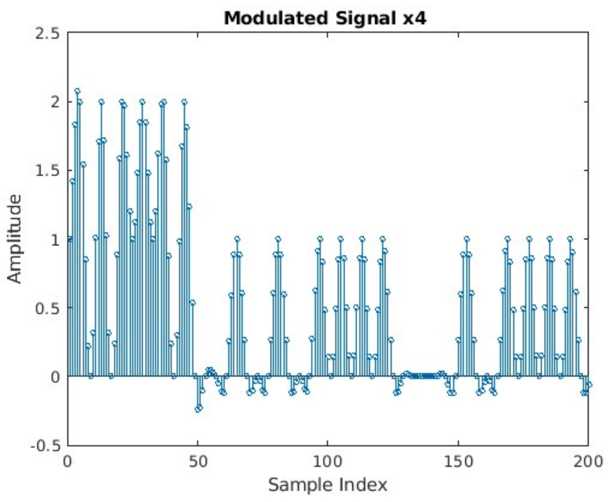

Fig. 5. 100 samples of modulated signal with rect function

Listing 4. modulator.m

```
function x4= modulator(x3,fc,Tb)

t = 0:Tb:(length(x3)-1)*Tb;
% t = t./fc;
carrier = cos(2*pi*fc*t)';

x4 = carrier.*x3;
end
```

VI. Channel and Noise Characteristics

The performance of a communication system significantly depends on the channel characteristics and the type of noise it encounters during signal transmission. In this project, we simulate the signal transmission through two types of Additive White Gaussian Noise (AWGN) channels to assess the robustness and effectiveness of our communication model under different noise conditions.

A. Memoryless AWGN Channel

The memoryless AWGN channel is a fundamental model used in communication theory to represent a channel with white noise. This model assumes that the noise added to the signal is Gaussian distributed, with zero mean and a constant power spectral density, independent of the transmitted signal. The mathematical model for the memoryless AWGN channel is given by:

Listing 5. memoryless Awgn Channel.m

```
function y4 = memoryless_awgn_channel(x4,
    EbN0_dB,txfil)

% y4=awgn(x4, EbN0_dB );

EbN0Lin = 10^(EbN0_dB/10);
Es = norm(txfil,2);
Eb = Es/4;
N0 = Eb/EbN0Lin;
sigma = sqrt(N0/2);
WGN = sigma*randn(size(x4));
y4 = x4 + WGN;


end
```

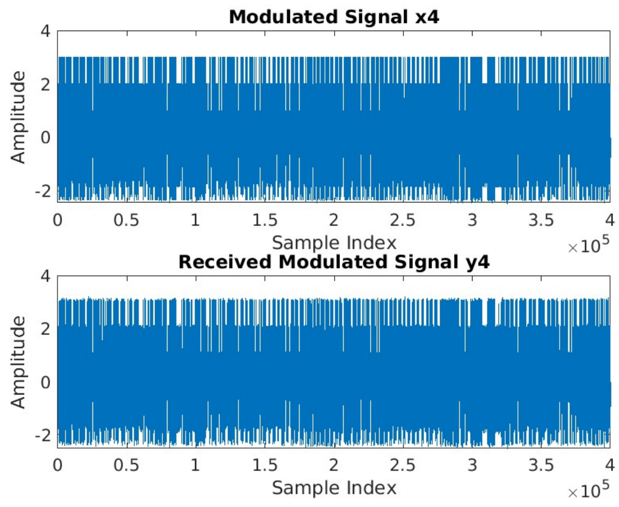

Fig. 6. Channel distortion to raised cosine modulated pulse by memoryless channel

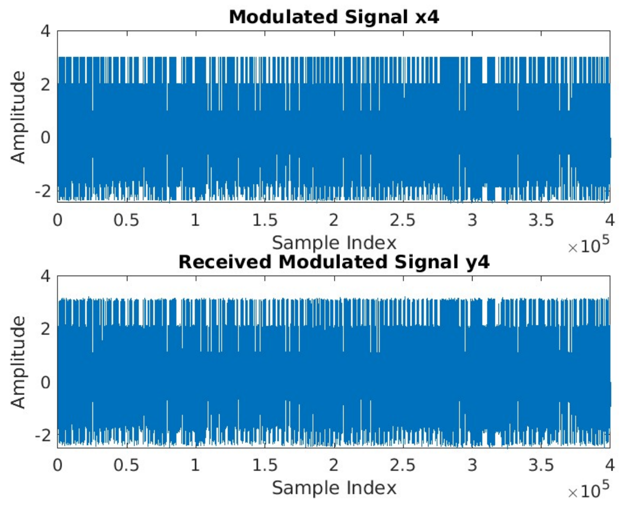

Fig. 7. Channel distortion to rect modulated pulse by memoryless channel

$$r(t) = s(t) + n(t)$$

where $s(t)$ represents the transmitted signal, $n(t)$ denotes the white Gaussian noise, and $r(t)$ is the received signal. This type of channel is used to model environments where the noise is random and uncorrelated with the signal or time, providing a baseline for evaluating communication system performance.

B. AWGN Channel with Memory

Unlike the memoryless model, an AWGN channel with memory introduces correlation in the noise, simulating more realistic scenarios where the channel's past affects the current noise characteristics. This can represent physical phenomena like multipath fading or echo. The channel model with memory is defined by:

$$r(t) = h(t) * s(t) + n(t)$$

where $h(t)$ is the channel impulse response, and $*$ denotes convolution. The impulse response used in this project is:

$$h(t) = a\delta(t) + (1-a)\delta(t - bT_b)$$

with a and b being parameters that adjust the memory effect of the channel, and δ representing the Dirac delta function. This model helps in studying the effects of temporal correlation within the channel on the signal integrity and overall system performance.

Listing 6. memory awgn channel.m

```
function y = memory_awgn_channel(x, EbN0_dB,
    a, b,Tb,txfil)


  A = 1;
  B = zeros(1,b*Tb);
  B(1) = a;
  B(end) = (1-a);

  % Filter the input signal through the channel
  x4 = filter(B, A, x);

  EbN0Lin = 10^(EbN0_dB/10);
  Es = norm(txfil,2);
  Eb = Es/4;
  N0 = Eb/EbN0Lin;
  sigma = sqrt(N0/2);
  WGN = sigma*randn(size(x4));
  y = x4 + WGN;
end
```

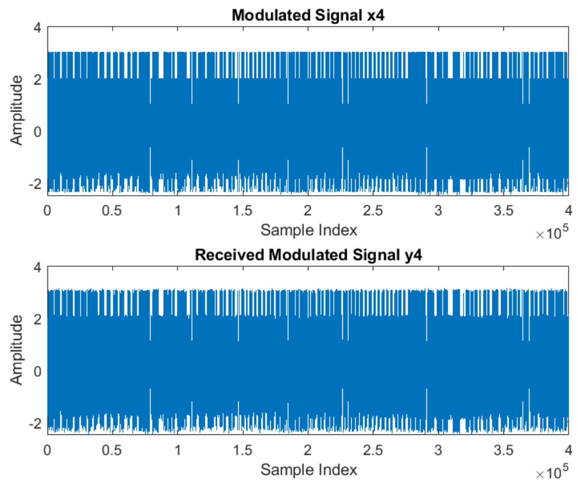

Fig. 8. Channel distortion to raised cosine modulated pulse by a channel with memory

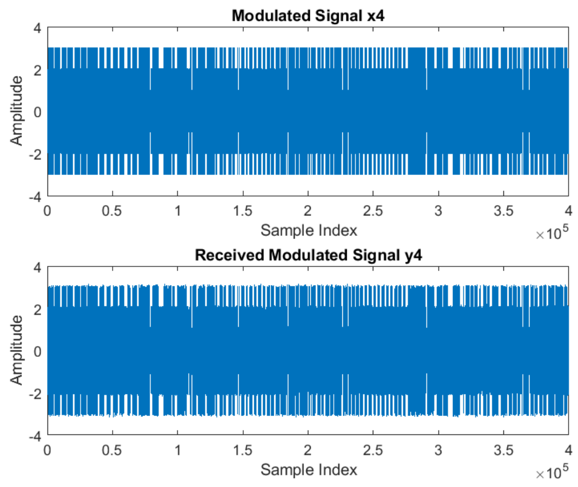

Fig. 9. Channel distortion to rect modulated pulse by a channel with memory

C. Simulation and Analysis

Simulations are conducted by transmitting the modulated signal through both channel types and analyzing the received signals. These simulations help in understanding how different channel conditions affect the error rates, signal-to-noise ratios, and the reliability of the communication system. Plots of the input and output signals for both channels, as well as their Power Spectral Densities (PSDs), are provided to visually assess the impact of each channel type on the communication quality

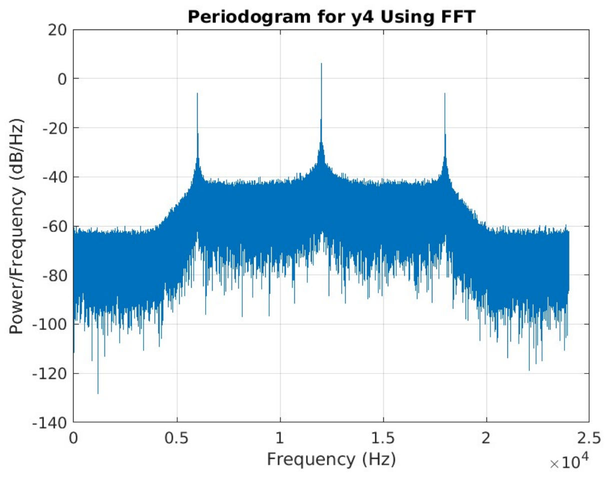

Fig. 10. PSD of raised cosine modulated pulse in memoryless channel

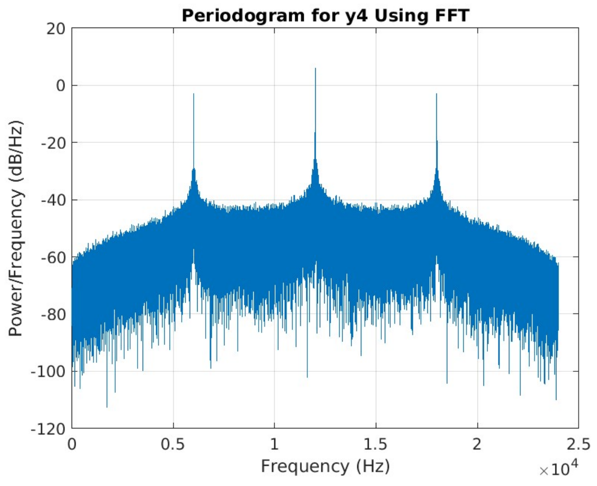

Fig. 11. PSD of rect modulated pulse by memoryless channel

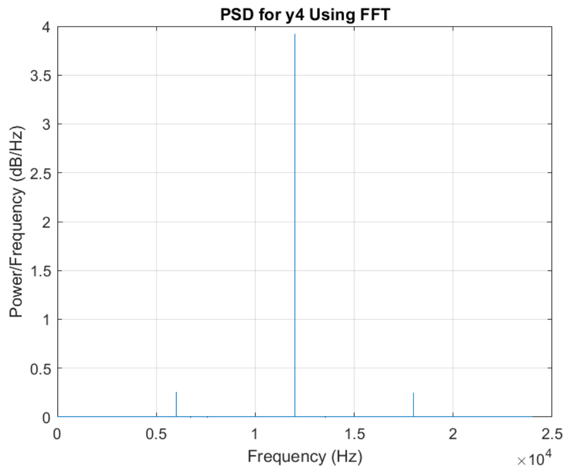

Fig. 12. PSD of raised cosine modulated pulse in memoryless channel

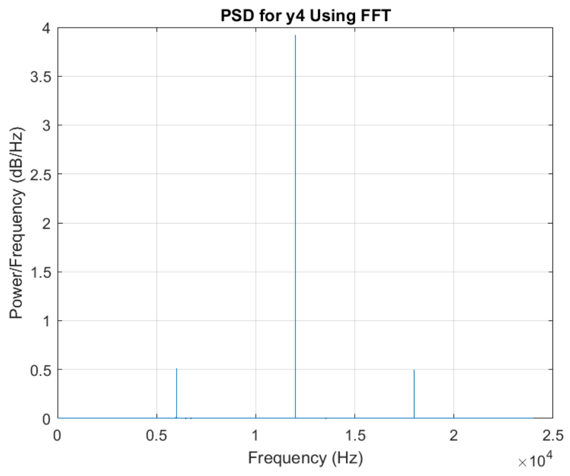

Fig. 13. PSD of rect modulated pulse by memoryless channel

In conclusion, the exploration of these two channel models enables a comprehensive understanding of how noise and memory in a channel can influence the performance of digital communication systems. This analysis is crucial for designing robust communication strategies that can effectively mitigate adverse channel effects and enhance signal integrity and reliability in real-world scenarios.

VII. Demodulation Process

Demodulation is the process of extracting the original information-bearing signal from a modulated carrier wave. In this project, we utilize the Hilbert transform for demodulation, a method particularly effective in the context of amplitude modulated signals like those used in our ASK scheme.

A. Hilbert Transform in Demodulation

The Hilbert transform is a widely used technique in signal processing to derive the analytic representation of a real-valued signal. It allows us to create a complex signal where the real part is the original signal, and the imaginary part is the Hilbert transform of the signal. This complex representation is crucial in various modulation techniques, especially those involving amplitude and phase variations.

In the context of our project, the Hilbert transform is used to demodulate the signal received from the channel, which contains both the desired signal and noise. The specific method of demodulation using the Hilbert transform can be described by the following MATLAB function:

Listing 7. demodulator.m

```
function y3 = demodulator(y4);
  %using hilbert filter
  y3 = abs(hilbert(y4));
end
```

Here, `y4` represents the signal received from the channel after it has passed through noise and other channel distortions. The function `hilbert(y4)` computes the Hilbert transform of `y4`, and the absolute value of this complex output provides the envelope of the original modulated signal, effectively demodulating it.

B. Role and Benefits

The use of the Hilbert transform in demodulating the signal offers several benefits:

- **Simplicity:** It simplifies the demodulation process by directly extracting the envelope of the modulated signal, which contains the original information.
- **Robustness:** It is robust against certain types of noise, particularly effective in systems where phase or frequency information is less critical.
- **Efficiency:** The Hilbert transform is computationally efficient, especially when implemented in environments like MATLAB that optimize such operations.

C. Analysis and Results

The effectiveness of this demodulation technique is demonstrated through the analysis of the demodulated signal `y3`. Plots comparing the original transmitted signal, the received noisy signal, and the demodulated output will be shown in the results section of this report. These visualizations will highlight the capability of the Hilbert transform to recover the original signal amidst noise, thereby validating its application in our communication system design.

D. Modulation Demodulation

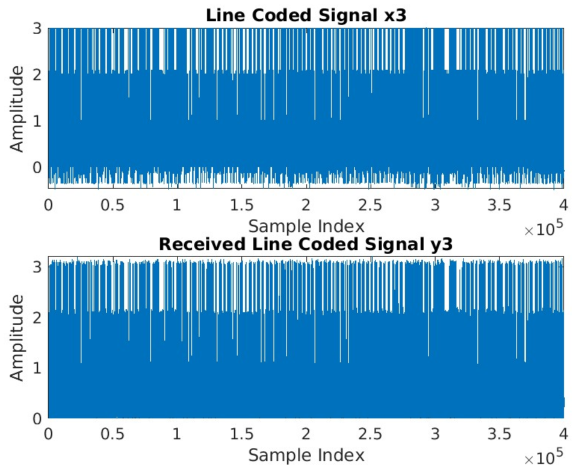

Fig. 14. demodulation of raised cosine modulated pulse in a memoryless channel

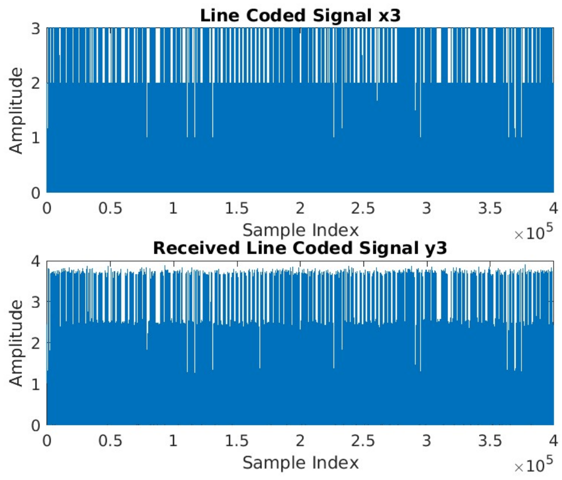

Fig. 15. demodulation of rect modulated pulse in a memoryless channel

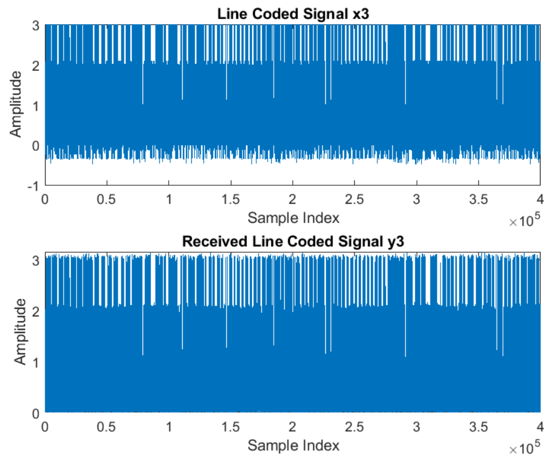

Fig. 16. demodulation of raised cosine modulated pulse in a channel with memory

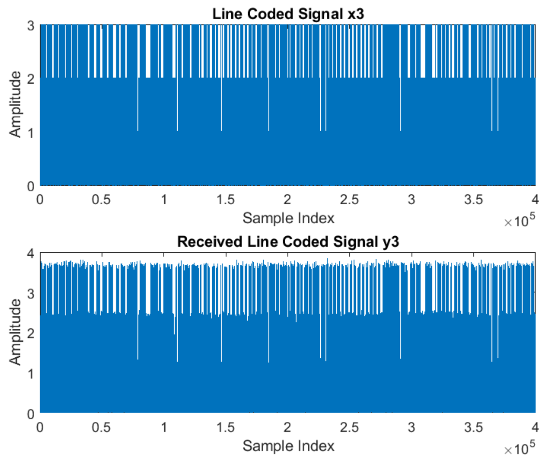

Fig. 17. demodulation of rect modulated pulse in a channel with memory

E. PSD Plots

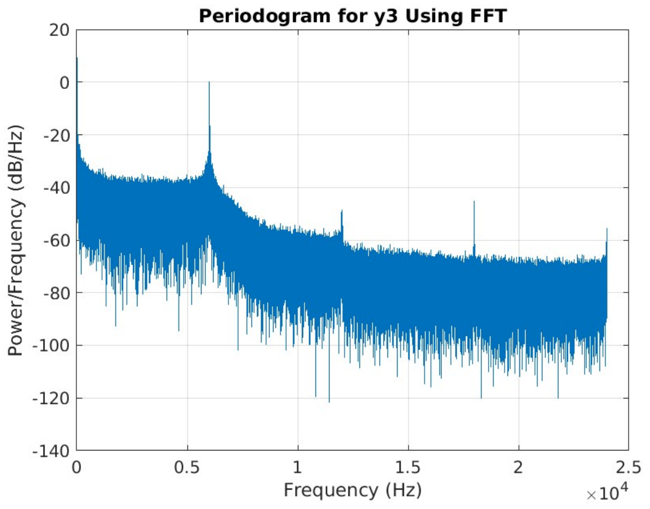

Fig. 18. PSD for received line decoded signal with raised cosine function through memoryless channel

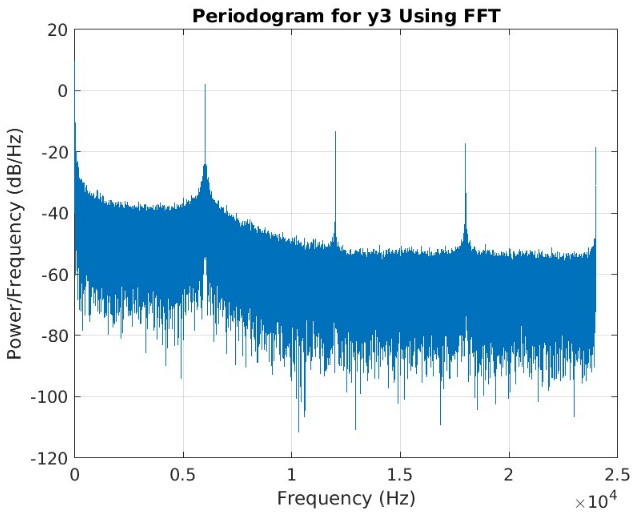

Fig. 19. PSD for received line decoded signal with rect function through memoryless channel

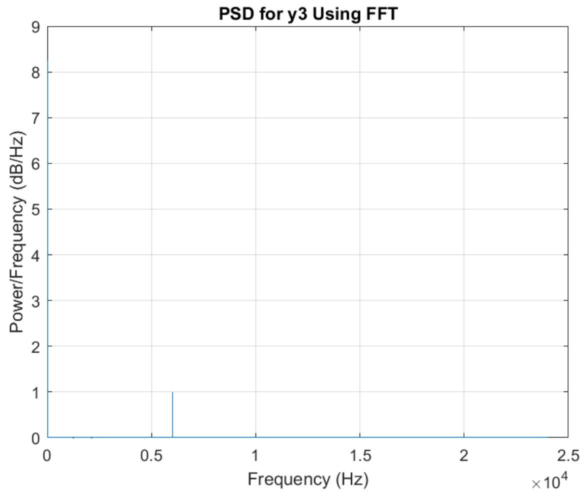

Fig. 20. PSD for received line decoded signal with rasied cosine function through channel with memory

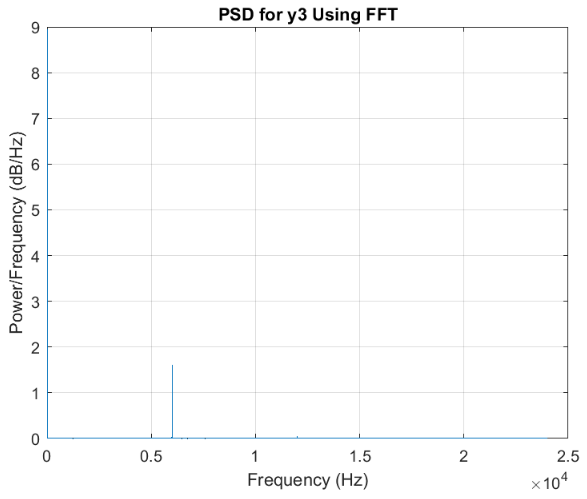

Fig. 21. PSD for received line decoded signal with rect function through channel with memory

In conclusion, the demodulation block is a pivotal component of our communication model. Using the Hilbert transform to demodulate the signal ensures that we can efficiently and accurately retrieve the transmitted information, which is crucial for the overall performance and reliability of the system.

VIII. Line Decoder Block

The line decoder block plays a crucial role in the digital communication system by reconstructing the digital signal from the demodulated analog signal. This step is crucial for preparing the data for subsequent decoding into the original information format. In our project, the line decoder simplifies the received signal by reducing the sampling rate that was increased during the encoding process, a technique known as downsampling.

A. *Function of the Line Decoder*

The primary function of the line decoder is to downsample the signal. Downsampling involves reducing the sample rate of the signal to its original rate or a rate suitable for further processing. This is necessary because the oversampling performed during earlier stages helps in combating issues like aliasing and provides room for better filtering but is not needed at the final stages of signal processing.

B. *Implementation of Line Decoder*

The MATLAB implementation of the line decoder utilizes a straightforward downsampling method where every m^{th} sample of the input signal is retained, and the others are discarded. This is performed by the function `lineDecoder`, defined as follows:

Listing 8. lineDecoder.m

```
function y2 = lineDecoder(y3,m)
  % y2 = conv(y3,ht,"same");
  y2 = y3(1:m:end);
end
```

In this function, `y3` is the input signal, which is the output from the demodulation block, and `m` represents the oversampling factor used during the modulation process. The line decoder selects every m^{th} sample of this signal, effectively reducing the sample rate by a factor of m. This reduction aligns the sample rate with the original rate at which the data was sampled or a rate appropriate for the final stages of digital signal processing.

C. *Role and Benefits*

The role of the line decoder in reducing the sample rate is pivotal for several reasons:

- **Efficiency:** It significantly reduces the amount of data that needs to be processed in subsequent stages, enhancing the overall efficiency of the system.
- **Error Reduction:** By matching the sampling rate to the original or a suitably lower rate, the likelihood of processing errors due to mismatched rates or excessive data is minimized.

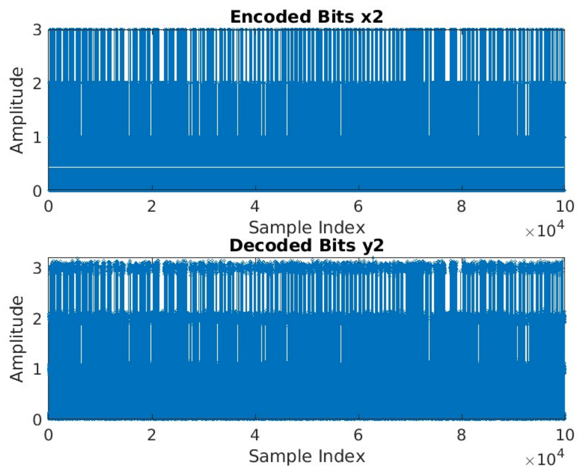

Fig. 22. Line decoding of raised cosine modulated pulse in a memoryless channel

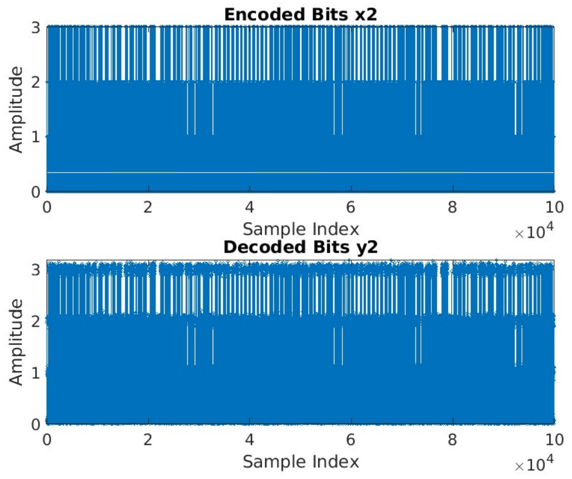

Fig. 23. Line decoding of rect modulated pulse in a memoryless channel

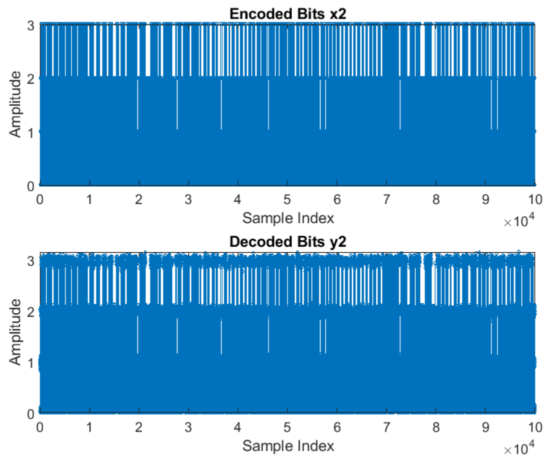

Fig. 24. Line decoding of raised cosine modulated pulse in a channel with memory

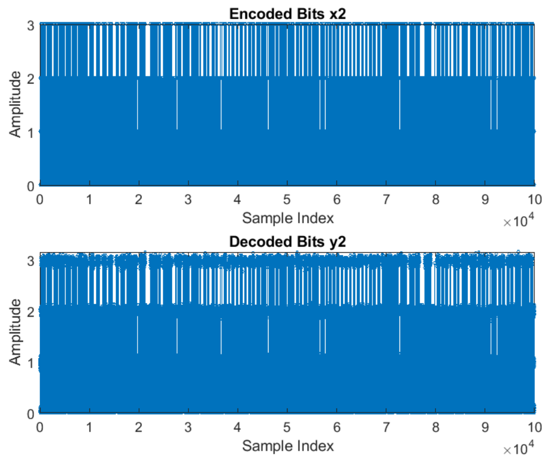

Fig. 25. Line decoding of rect modulated pulse in a channel with memory

D. Conclusion

The line decoder block is essential for converting the over-sampled analog signal back into a digitally manageable form, ready for final decoding and analysis. The effectiveness of this block is crucial for the overall performance and accuracy of the communication system, as it ensures that the signal is restored to a form most conducive to accurate information retrieval and further processing.

IX. Symbol Decoder Block

The final stage in our digital communication system is the symbol decoder block, which translates the processed analog signal back into digital form. This conversion is crucial for reconstructing the original digital data transmitted from the sender, effectively completing the communication cycle.

A. Function of the Symbol Decoder

The symbol decoder's primary function is to map the received analog signal values back into the corresponding digital bit sequences based on predefined threshold levels. This mapping is vital as it recovers the original digital information from the modulated and transmitted signal that has undergone various transformations and potential noise interference during transmission.

B. Implementation of Symbol Decoder

The MATLAB implementation of the symbol decoder employs a series of conditional statements to categorize the input signal into different digital states based on threshold values. These thresholds are determined by observing the signal constellation plots and identifying optimal boundaries that minimize error in symbol interpretation. The following MATLAB function illustrates this decoding process:

Listing 9. decoder.m

```
function y1 = decoder(y2)
  y1 = [];
  for i = y2'
    if(i>2.5)
      y1 = [y1;1;1];
    elseif (2.5>i & i>1.5)
      y1 = [y1;1;0];
    elseif (1.5>i & i>0.5)
      y1 = [y1;0;1];
    elseif (i<0.5)
      y1 = [y1;0;0];
    end
  end
end
```

This function, `decoder`, takes the line decoded signal `y2` as input. It processes each element of `y2`, classifying it into one of the four possible two-bit sequences based on its amplitude. The conditions within the loop check the amplitude against the thresholds:

- Values greater than 2.5 map to the binary sequence [1 , 1].
- Values between 1.5 and 2.5 map to [1 , 0].
- Values between 0.5 and 1.5 map to [0 , 1].
- Values less than 0.5 map to [0 , 0].

C. Role and Benefits

The decoding process is fundamental for:

- **Accuracy:** Ensuring the accuracy of the digital data recovered at the receiver end.
- **Error Minimization:** By carefully setting the thresholds based on the constellation diagram, the decoder minimizes the probability of bit errors introduced by channel noise and signal distortions.
- **Data Integrity:** It guarantees the integrity of the transmitted information, making sure that the data received is as close to the original as possible.

D. Recieved Bits Plots

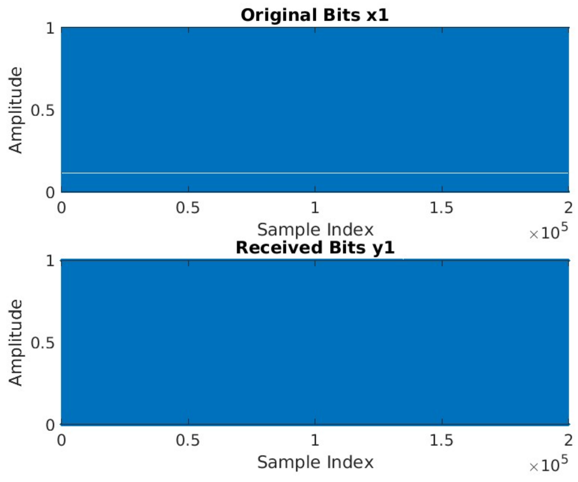

Fig. 26. Symbol decoding of raised cosine modulated pulse in a memoryless channel

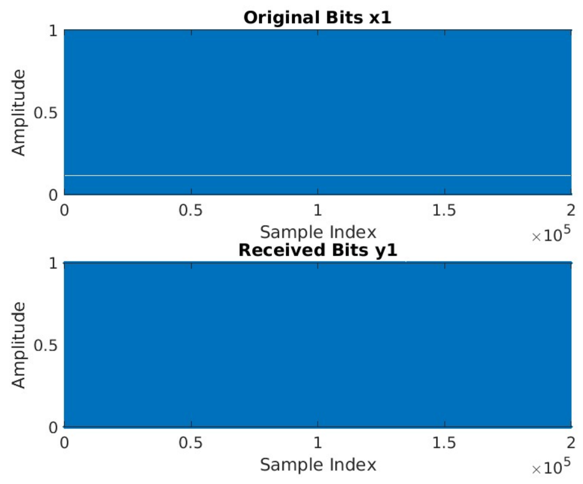

Fig. 27. Symbol decoding of rect modulated pulse in a memoryless channel

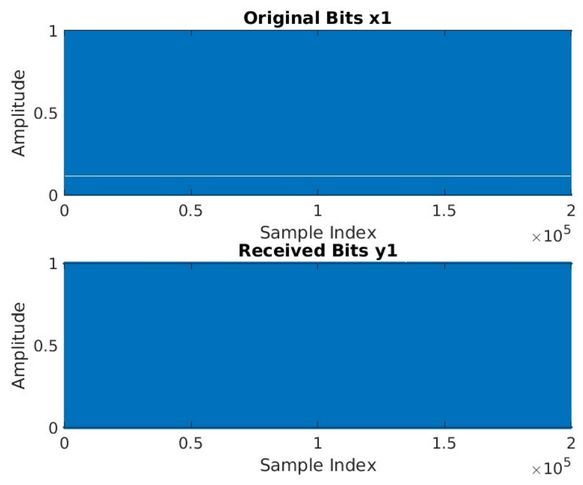

Fig. 28. Symbol decoding of raised cosine modulated pulse in a channel with memory

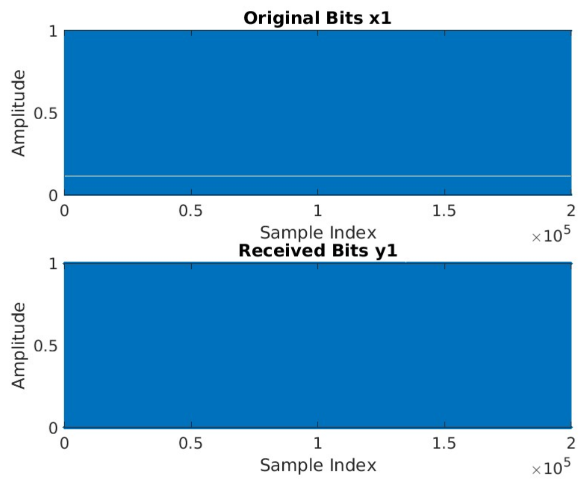

Fig. 29. Symbol decoding of rect modulated pulse in a channel with memory

E. BER vs SNR

In digital communication systems, the Bit Error Rate (BER) versus Signal-to-Noise Ratio (SNR) relationship is crucial, particularly in Additive White Gaussian Noise (AWGN) channels. AWGN channels model random noise effects with constant power across all frequencies and a Gaussian distribution. BER measures the ratio of erroneous bits to total transmitted bits, indicating system quality, while SNR represents signal strength relative to noise. In AWGN channels, higher SNR improves signal quality, lowering BER. BER vs SNR curves illustrate this relationship, with diminishing returns at high SNR. Error correction and detection techniques are vital for achieving low BER, often requiring trade-offs in bandwidth, power, or complexity. Understanding this interplay is essential for optimizing communication systems.

Memoryless Channel

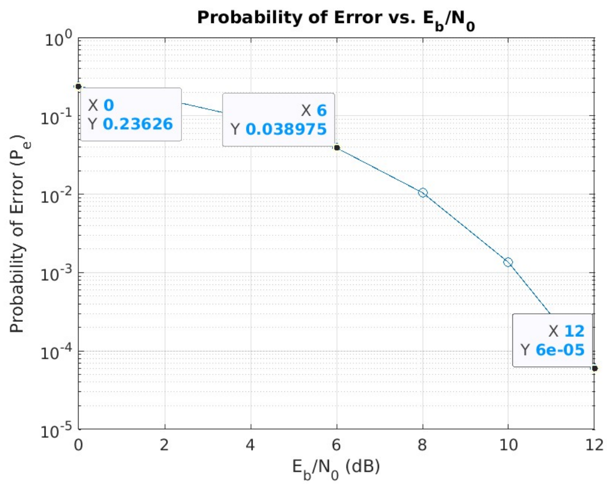

Fig. 30. P_e vs SNR for raised cosine pulse in memoryless channel

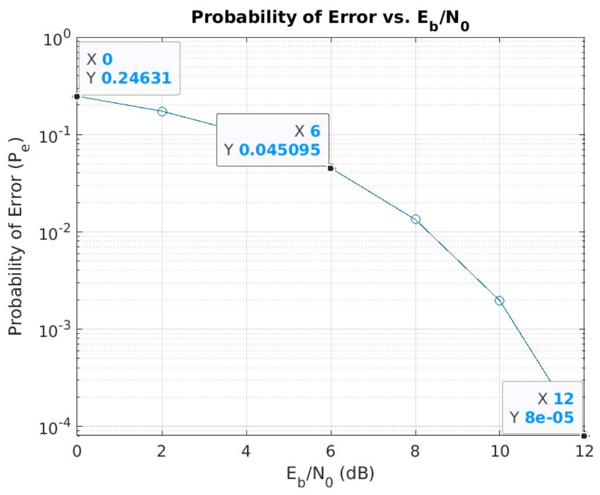

Fig. 31. P_e vs SNR for rect pulse in memoryless channel

Channel wih memory

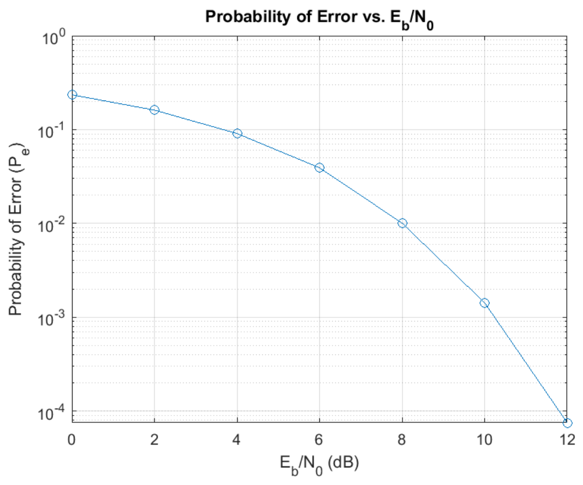

Fig. 32. P_e vs SNR for raised cosine pulse in channel with memory

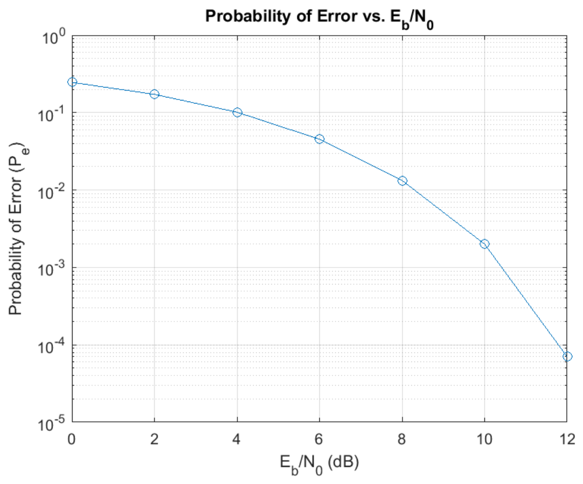

Fig. 33. P_e vs SNR for rect pulse in channel with memory

F. Output Constellations

The constellation of a signal can be though of all the vectors used to represent the decoded symbols that are plotted in the complex plane. The resulting diagram that emerges from plottingnthese vectors is called the constellation. The constellation for our system for different Noise levels are shown as follows:

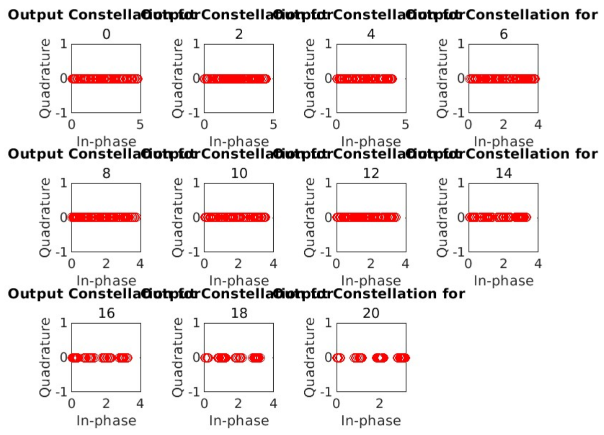

Fig. 34. Constellation of raised cosine pulse in memoryless channel for snr from 0 to 20

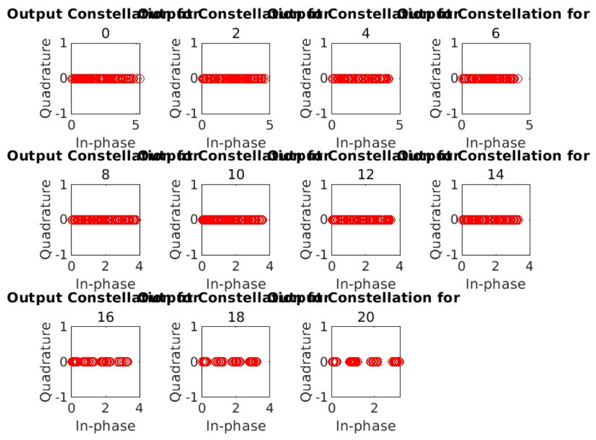

Fig. 35. Constellation of rect pulse in memoryless channel for snr from 0 to 20

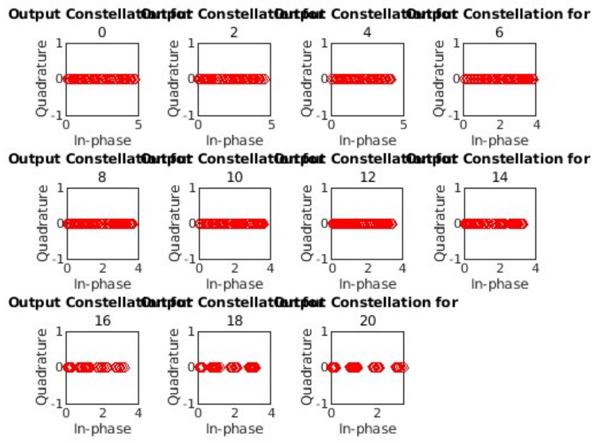

Fig. 36. Constellation of raised cosine pulse in channel with memory for snr from 0 to 20

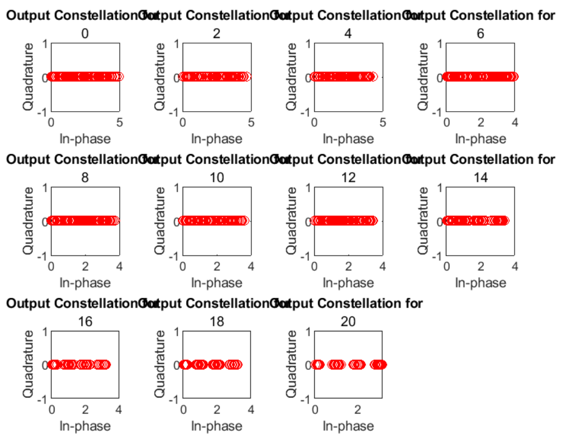

Fig. 37. Constellation of rect pulse in channel with memory for snr from 0 to 20

G. Conclusion

The symbol decoder block is a critical component of the communication system, directly impacting the effectiveness and reliability of the data transmission process. By converting analog signal values back into digital bit sequences, the decoder ensures that the information conveyed through the communication system is accurately reconstructed, maintaining the fidelity and integrity of the original data.

X. Digital-to-Analog Converter Block

The Digital-to-Analog Converter (D/A Converter) block is essential in digital communication systems, especially when the end goal is to reconstruct audio signals from digital data. In our project, this block performs the crucial task of converting the processed digital bits back into an analog audio format, which can be played back or used in further applications.

A. Function of the D/A Converter

The D/A converter's primary function is to transform digital binary data into a continuous analog signal. This process involves several steps, from interpreting binary sequences as numerical values to converting these values into a physical signal that accurately represents the original audio content.

B. Implementation of the D/A Converter

The MATLAB implementation of the D/A converter is designed to handle a stream of binary data, which represents the decoded digital audio information. The following MATLAB function, `D_A_converter`, illustrates the conversion process from binary data back to audio:

In this function:

- `y1` is the input array of binary digits representing the decoded signal.
- `filename` is the name of the output file where the audio will be saved.
- `Fs` is the sampling frequency of the audio signal.

The binary data is first segmented into 8-bit chunks, which are then converted from binary to decimal values and subsequently cast into an unsigned 8-bit integer format. These integers are then typecast to single-precision floating-point numbers, compatible with MATLAB's audio processing functions. The audio data is assumed to be stereo, hence reshaped into two channels before it is finally written to an audio file using `audiowrite`.

C. Role and Benefits

The D/A conversion process is critical for:

- **Signal Reconstruction:** It allows the accurate reconstruction of the original audio signal from its digital representation, essential for auditory analysis and playback.
- **Data Verification:** Enables the verification of the integrity and accuracy of the communication process by comparing the original and reconstructed audio signals.
- **Practical Applications:** Facilitates the practical application of theoretical digital communication concepts in real-world scenarios, such as telecommunications and broadcasting.

Listing 10. D A converter.m

```
function D_A_converter(y1,filename,Fs)

 binData = reshape(y1,length(y1)/8,8);
 uint8Data =
     uint8(bin2dec(char(binData+'0')));
 audioData = typecast(uint8Data, 'single');
 numSamples = size(audioData, 1) / 2; %
     Assuming two channels
 audioData = reshape(audioData, numSamples,
     2);
 audiowrite(filename, audioData,Fs);

end
```

D. Conclusion

The Digital-to-Analog converter block is a vital component of our communication system, ensuring that the digitally transmitted information is converted back into an analog format that can be utilized in practical applications. This conversion is crucial for the effectiveness and utility of the entire digital communication system, enabling the accurate playback and analysis of transmitted audio signals.