Communication Theory Project Report

Course Code:	EC5.203
Term:	Spring Semester 2024
Project:	QPSK Communication Model
Team:	Koustubh Jain, 2023122003 koustubh.jain@research.iiit.ac.in Soumil Gupta, 2022102035 soumil.gupta@students.iiit.ac.in
Date:	9th May 2024

1 Overview

This project aims to simulate a QPSK Communication Model, with the transmitter side composed of an A/D converter, Encoder which maps the bits to QPSK values with a $\pi/4$ phase shift, a line coder which converts the binary data to actual baseband waveforms for transmission, a modulator which modulates the baseband signal to a carrier frequency of 1 MHz and the corresponding reverse blocks on the receiver side, so as to recover the original audio signal that was transmitted with minimal noise.

2 A/D Converter

The A/D converter converts the given analog `.wav` file into a digital sequence. We first use the function `audioread()` and extract mono audio from the given stereo audio. After this, depending on a fixed precision value (in our case upto 4 decimal places) the value is normalised and then rounded off to the nearest integers. Following which it is represented in binary using minimum possible number of bits (in our case 14).

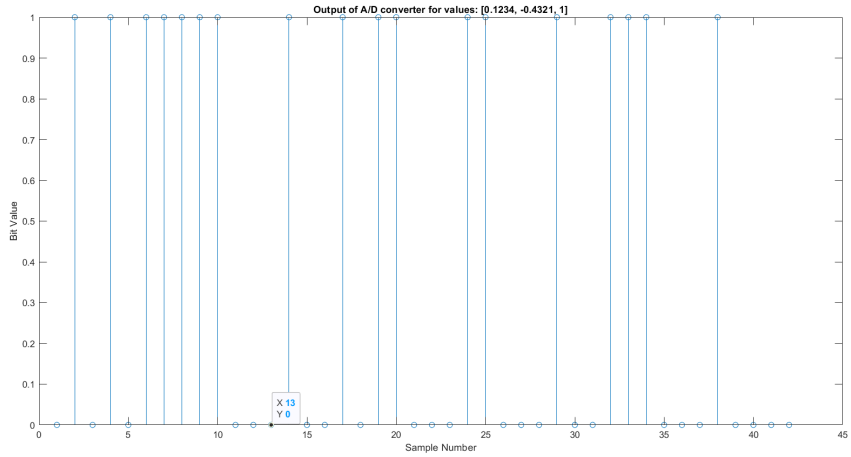

Figure 1: A/D converter output

3 Encoder

The encoder block maps the incoming bitstream from the A/D converter as per the QPSK constellation shifted counterclockwise by $\pi/4$ i.e. it takes to

two consecutive bits and maps them to appropriate values. For example : $(0,0) \rightarrow (a,a); (1,0) \rightarrow (-a,a); (0,1) \rightarrow (a,-a); (1,1) \rightarrow (-a,-a)$

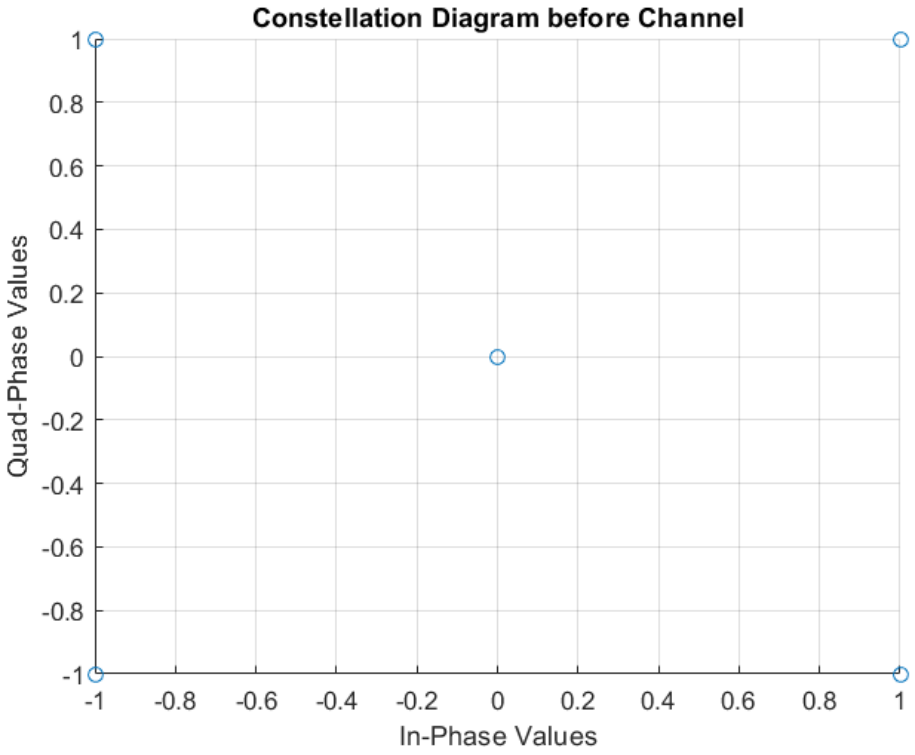

Figure 2: QPSK Mapping

4 Line Coding

Using the line coding block, we convert the binary data generated by encoder into actual baseband waveforms which can be further modulated for transmission. The output of the line coder is given as follows:

$$x_3(t) = \Sigma_k a_k p(t - kT_b)$$

where a_k are the encoded bits and $p(t-kTb)$ is a matched filter to the desired pulse waveform i.e. either a raised cosine pulse or a rectangular pulse.

For the raised cosine pulse, the following parameters have been chosen :

1. Roll-off factor = 1 ; it affects the steepness of the slope
2. Samples per symbol = 4

3. Length of the pulse = 17 ; encoded data is up-sampled by the filter length to prepare it for pulse-shaping by increasing sampling rate.

Similarly, the rectangular pulse is a boxcar function of length 17. The input signal is simply convolved with this pulse after up-sampling, to get the line coded signal.

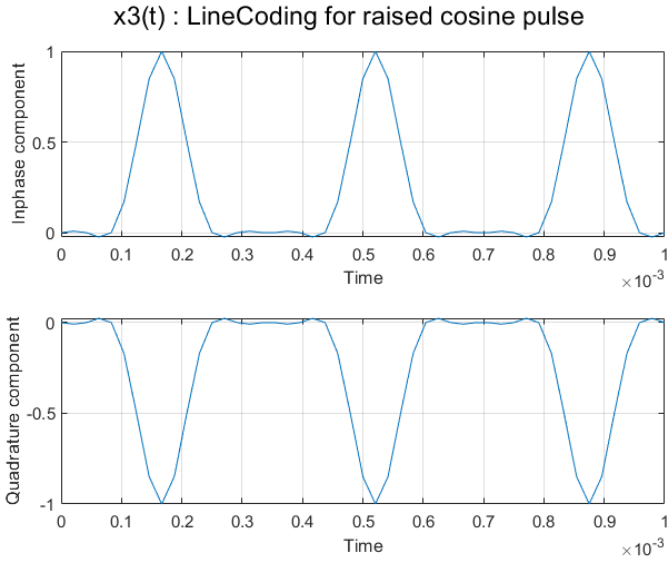

Figure 3: Raised Cosine Line Coded Signal

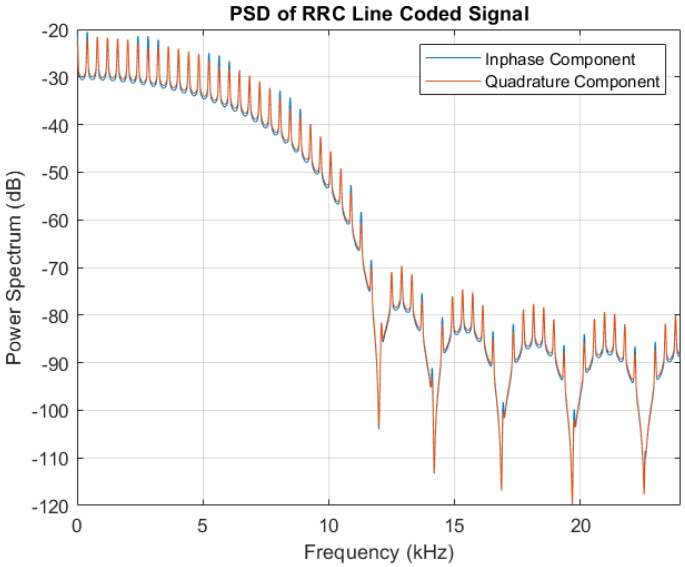

Figure 4: PSD of Raised Cosine Line Coded Signal

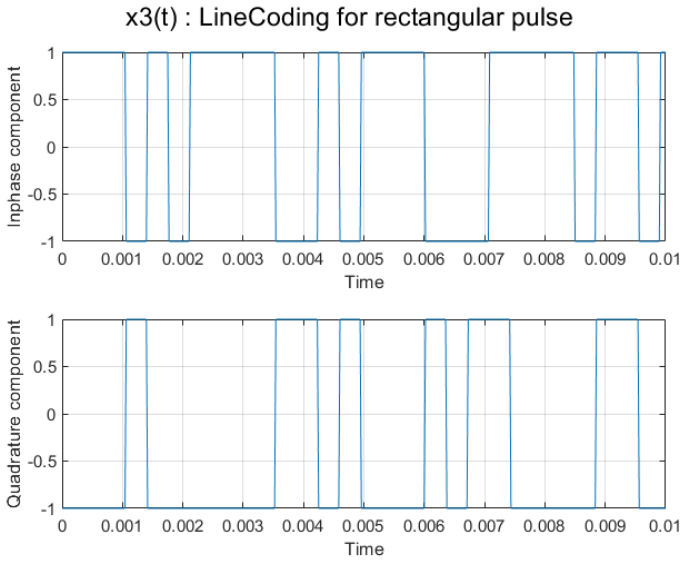

Figure 5: Rectangular Line Coded Signal

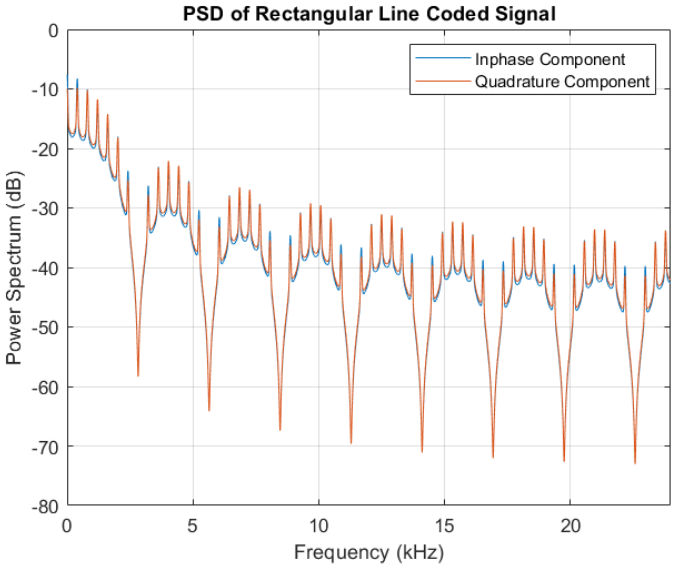

Figure 6: PSD of Rectangular Line Coded Signal

5 Modulation

The QPSK modulated signal denoted by $\phi_{PSK}(t)$ is given by the following equation

$$\phi_{PSK}(t) = a_m \sqrt{\frac{2}{T_b}} \cos \omega_c t + b_m \sqrt{\frac{2}{T_b}} \sin \omega_c t$$

where a_m and b_m are the in-phase and quadrature bits fed into the modulator block from the line coding block.

The input to the modulator is a 1x2 cell array, where the first array inside the cell is the inphase bitstream (a_m) and the second array inside the cell, is the quadrature bitstream (b_m).

Since our constellation is shifted by $\frac{\pi}{4}$, the expression for the QPSK signal becomes:

$$\phi_{PSK}(t) = a_m \sqrt{\frac{2}{T_b}} \cos\left(\omega_c t + \frac{\pi}{4}\right) + b_m \sqrt{\frac{2}{T_b}} \sin\left(\omega_c t + \frac{\pi}{4}\right)$$

In our implementation of QPSK, since our filter length is 17, hence $T_b = 17$ and given a carrier frequency of 1 MHz, the equation for the QPSK

modulated waveform resolves to:

$$\phi_{PSK}(t) = a_m\sqrt{\frac{2}{17}}\cos\left(2\pi\cdot 10^6\cdot t+\frac{\pi}{4}\right)+b_m\sqrt{\frac{2}{17}}\sin\left(2\pi\cdot 10^6\cdot t+\frac{\pi}{4}\right)$$

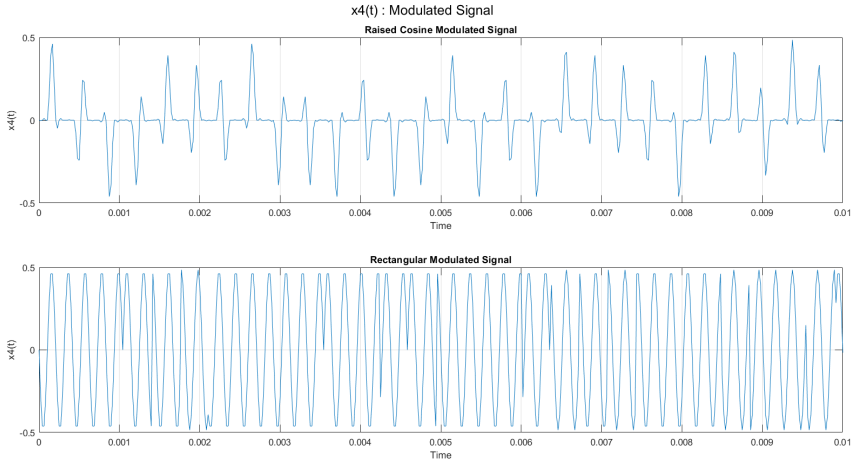

Figure 7: $x_4(t)$:Output of Modulator

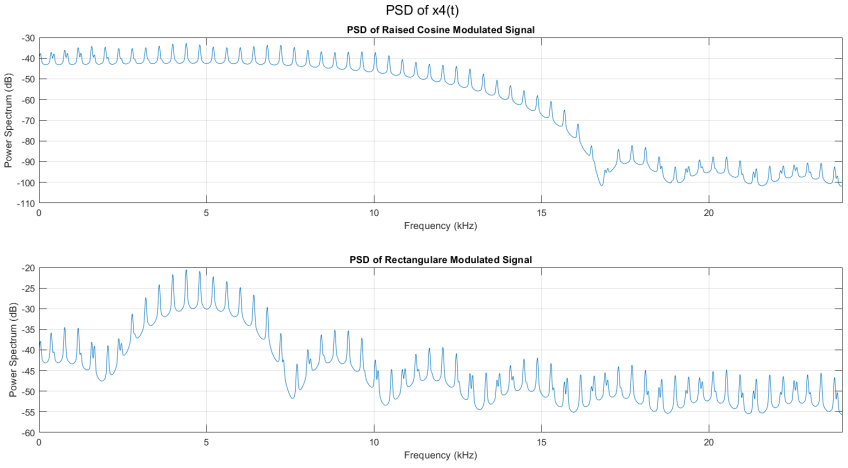

Figure 8: PSD of $x_4(t)$

6 Channel

The modulated signal is then passed through an AWGN channel. For the memoryless channel, we simply generate random noise values from the zero-mean standard normal distribution and add it to the modulated signal. For the channel with memory, we convolve the transmitted signal with a pulse train. This pulse train has two pulses, whose energy distribution can be controlled with the parameters `a` and `b`.

$$r_{\text{memoryless}}(t) = s(t) + n(t)$$
$$r_{\text{with memory}}(t) = h(t) * s(t) + n(t)$$

where $h(t) = a\delta(t) + (1-a)\delta(t - bT_b)$ and $s(t)$ is the transmitted signal and $n(t)$ is the additive white Gaussian noise.

6.1 Comparative Analysis of AWGN Channels

1. The Memory-less AWGN channel introduces a time-invariant and constant additive noise across the entire spectrum whereas the AWGN channel with memory is called as such, because the noise introduced by it, is time dependent.

2. The sum total of the Gaussian noise and the convolution with the impulse response make this noise distribution non-Gaussian.

3. The Memory-less channel treats each symbol independently whereas the channel with memory introduces correlations, which may affect symbol integrity.

4. Memoryless channels generally tend to have lower P_e as compared to channels with Memory.

5. Memoryless channel requires less complex receiver as compared to channel with memory.

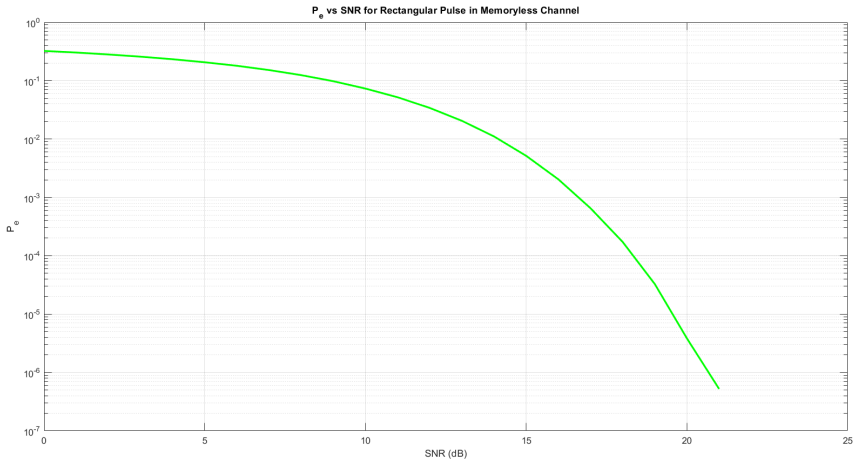

Figure 9: P_e v/s SNR for memoryless channel

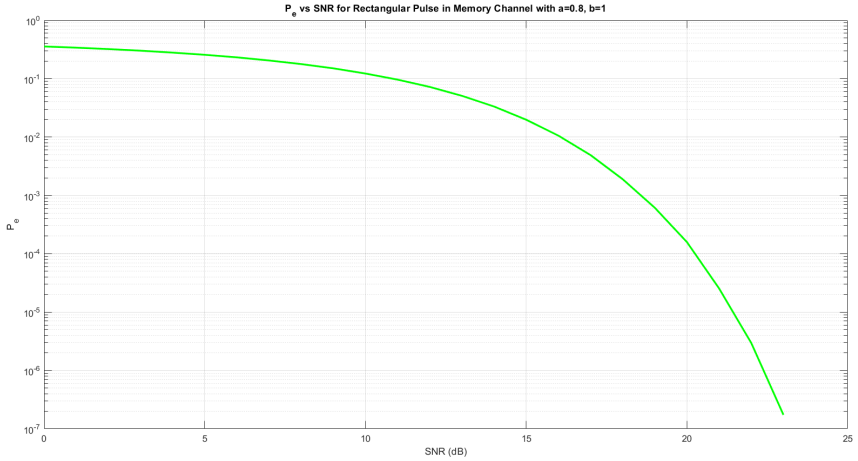

Figure 10: P_e v/s SNR for channel with memory

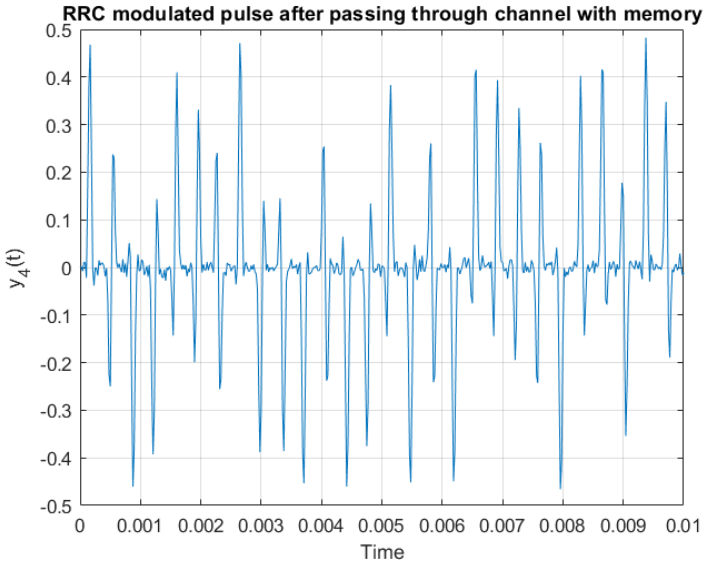

Figure 11: RC Modulated Pulse after passing through channel with memory with $a = 1$ and $b = 1$

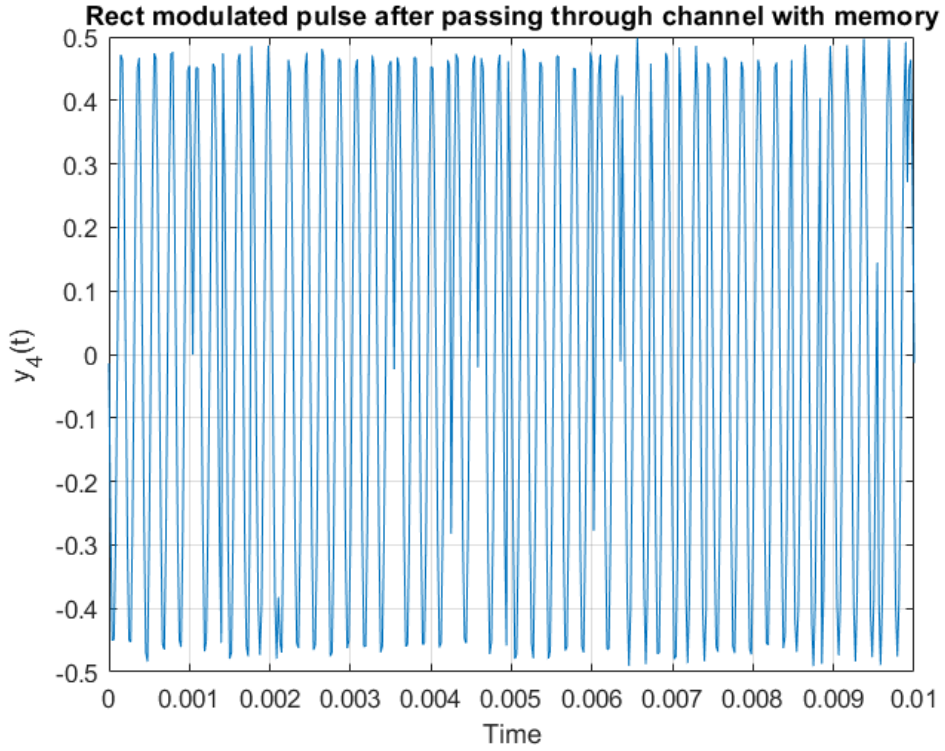

Figure 12: Rect Modulated Pulse after passing through channel with memory with $a = 1$ and $b = 1$

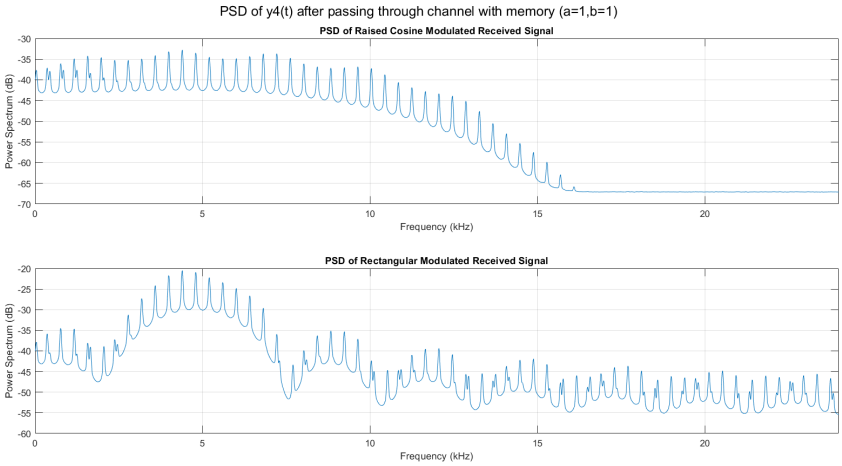

Figure 13: PSD of $y_4(t)$ after passing through channel with memory with $a = 1$ and $b = 1$

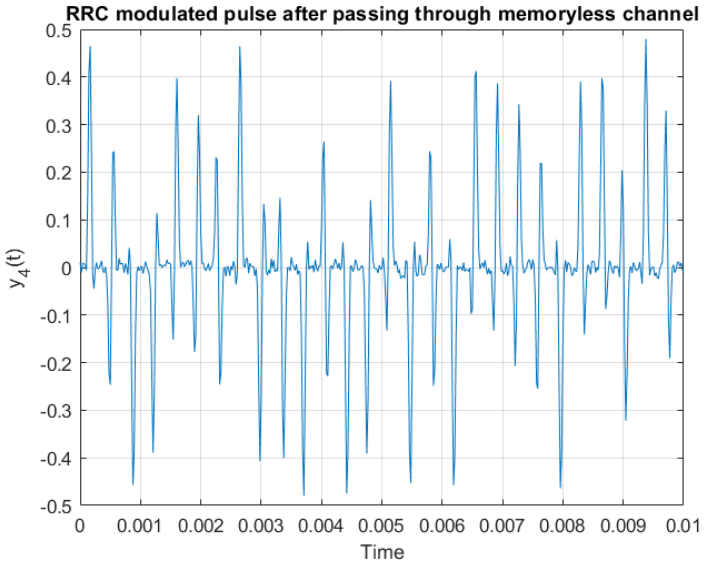

Figure 14: RC Modulated Pulse after passing through memory-less channel

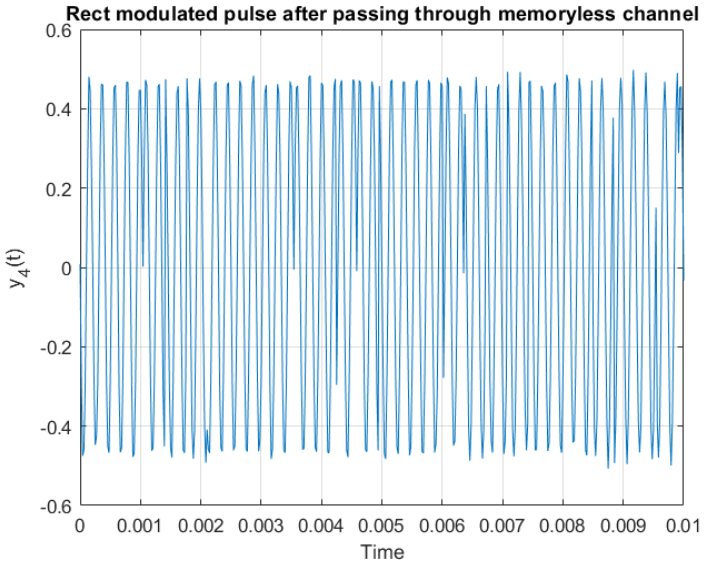

Figure 15: Rect Modulated Pulse after passing through memory-less channel

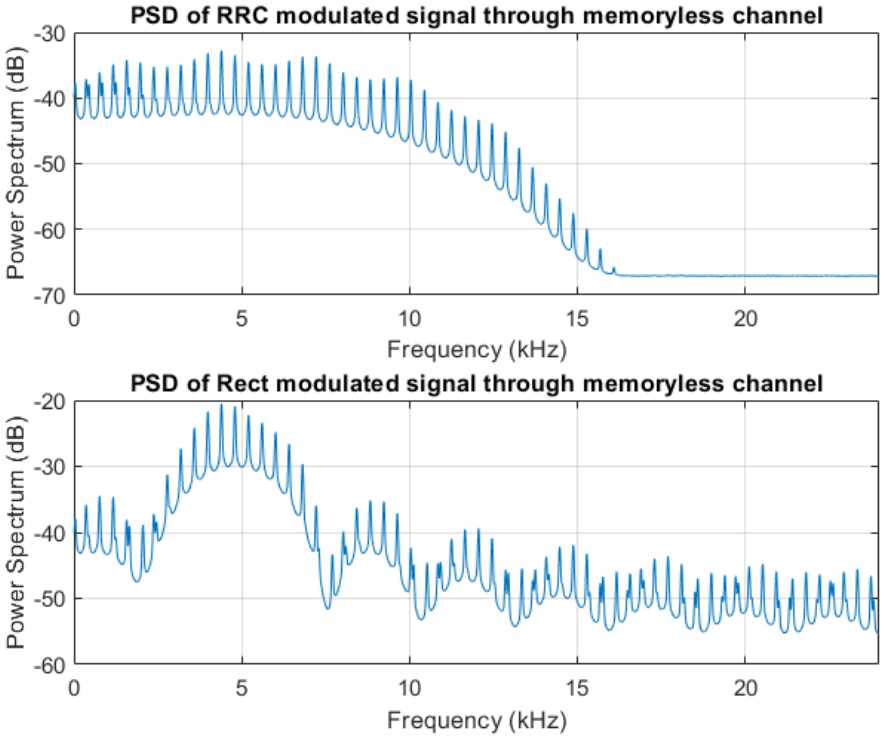

Figure 16: PSD of $y_4(t)$ after passing through memoryless channel

7 Demodulation

To demodulate the received signal and extract the inphase and quadrature bit streams separately from it, we multiply the incoming signal with its respective carrier and then proceed to low pass filter it. Since, the two bit streams were modulated using carriers which are also in quadrature, they will be removed by low pass filtering. Finally we will be able to extract the inphase bits and quadrature bits with some remaining channel noise.

For example, for the memory-less channel model,In order to extract a_m with some remaining channel noise

$$\begin{aligned} r(t) &= \left(a_m\sqrt{\frac{2}{T_b}}\cos\left(\omega_c t+\frac{\pi}{4}\right)+b_m\sqrt{\frac{2}{T_b}}\sin\left(\omega_c t+\frac{\pi}{4}\right)+n(t)\right)\times\cos\left(\omega_c t+\frac{\pi}{4}\right) \\ &= a_m\sqrt{\frac{2}{T_b}}\cos\left(\omega_c t+\frac{\pi}{4}\right)^2+\frac{b_m}{2}\sqrt{\frac{2}{T_b}}\sin\left(2\omega_c t+\frac{2\pi}{4}\right)+n(t)\cos\left(\omega_c t+\frac{\pi}{4}\right) \\ &= \frac{a_m}{2}\sqrt{\frac{2}{T_b}}\left(1+\cos\left(2\omega_c t+\frac{2\pi}{4}\right)\right)+\frac{b_m}{2}\sqrt{\frac{2}{T_b}}\sin\left(2\omega_c t+\frac{2\pi}{4}\right)+n(t)\cos\left(\omega_c t+\frac{\pi}{4}\right) \end{aligned}$$

When this is now low pass filtered, the only remaining term is a_m times some scaling factor and the noise modulated with the carrier frequency, which we can reduce but not get rid of completely.

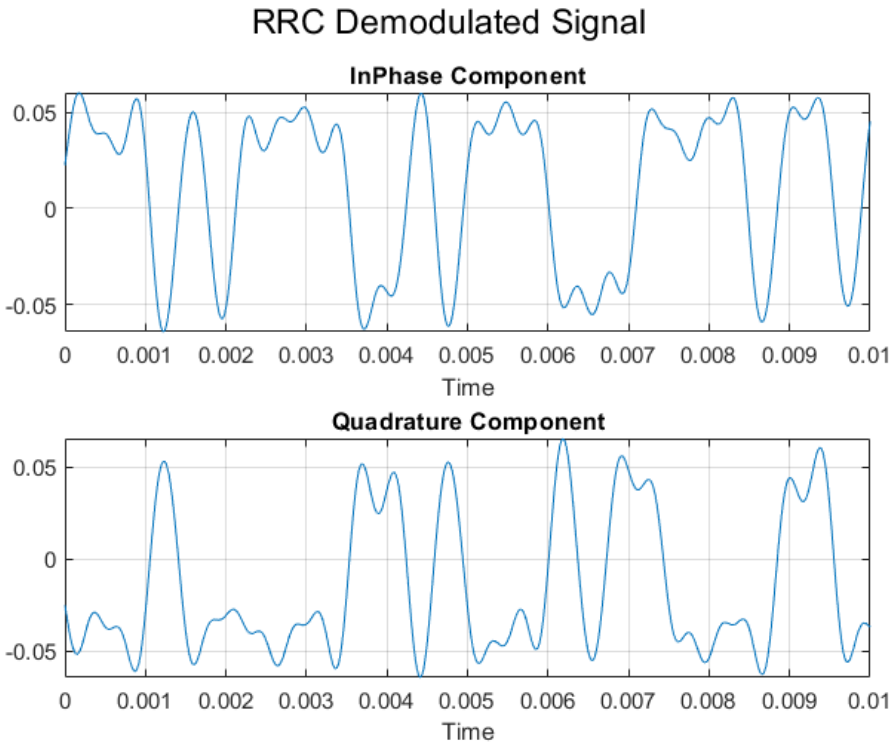

Figure 17: RC Modulated waveform after passing through channel with memory(a=1,b=1) and after demodulation

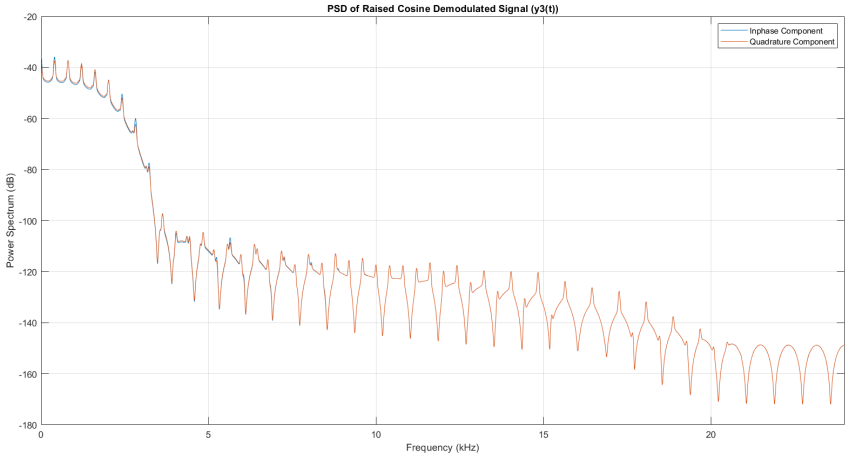

Figure 18: PSD of $y_3(t)$ when incoming signal was RC modulated and passed through channel with memory (a=1,b=1)

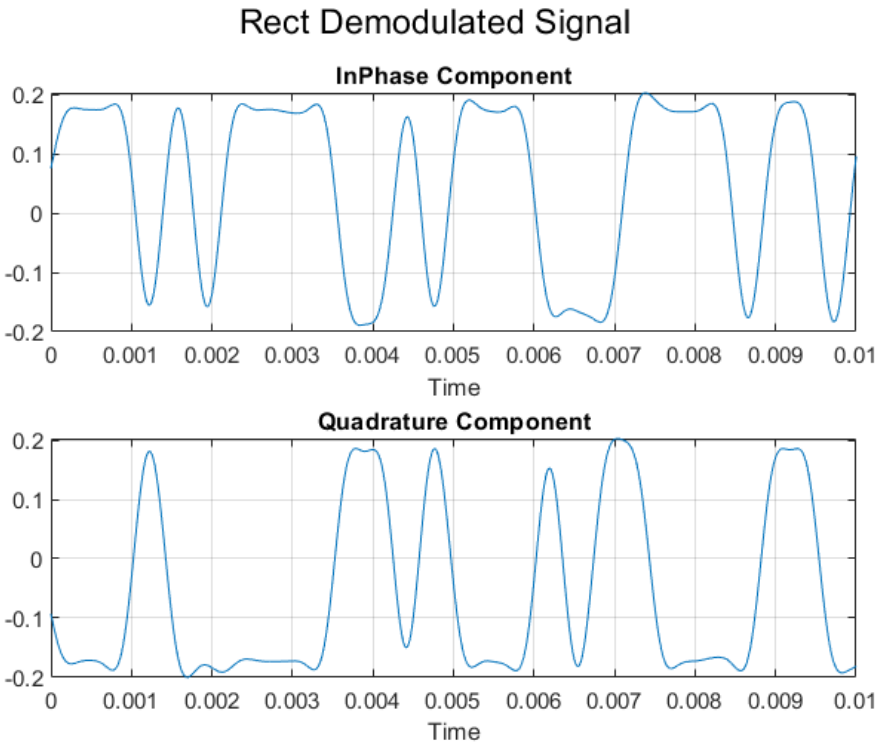

Figure 19: Rect Modulated waveform after passing through channel with memory(a=1,b=1) and after demodulation

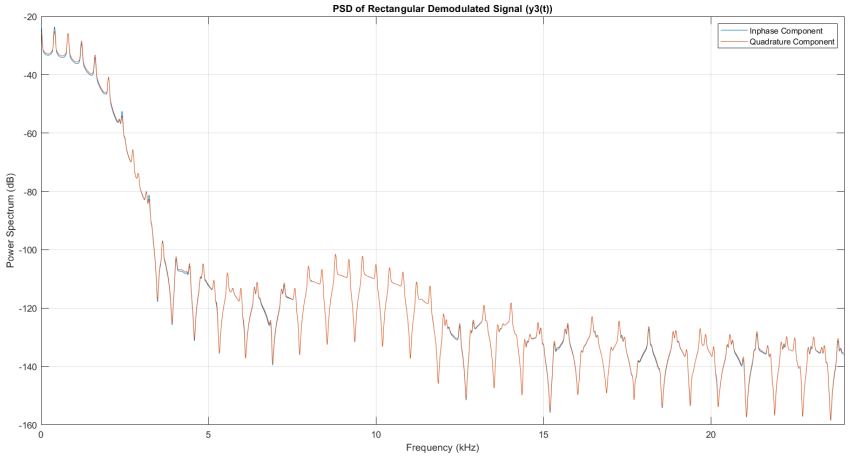

Figure 20: PSD of $y_3(t)$ when incoming signal was Rect modulated and passed through channel with memory (a=1,b=1)

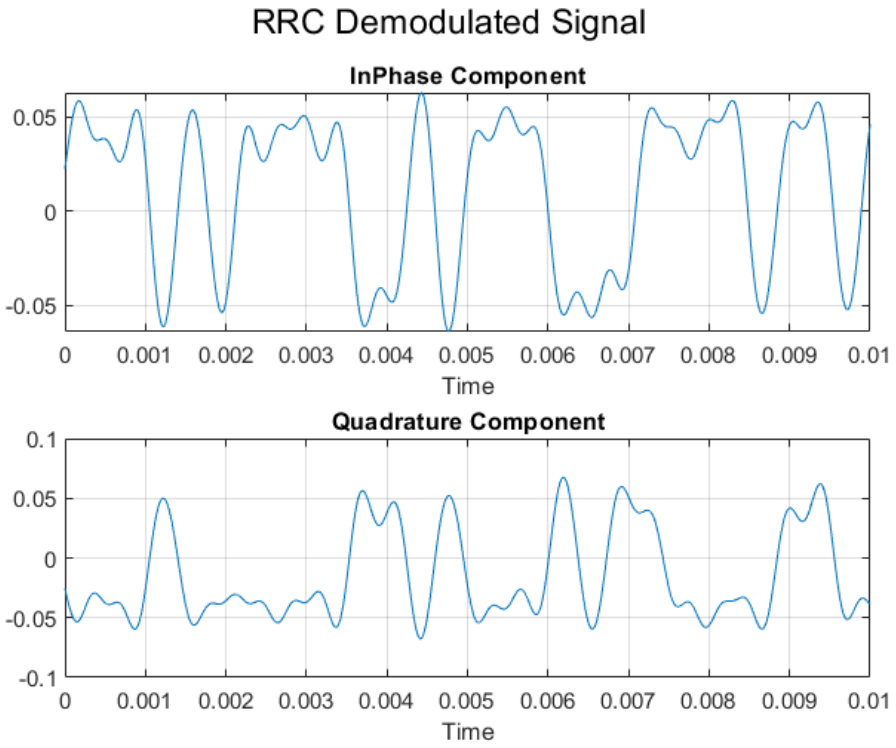

Figure 21: RC Modulated waveform after demodulation and after memory-less channel

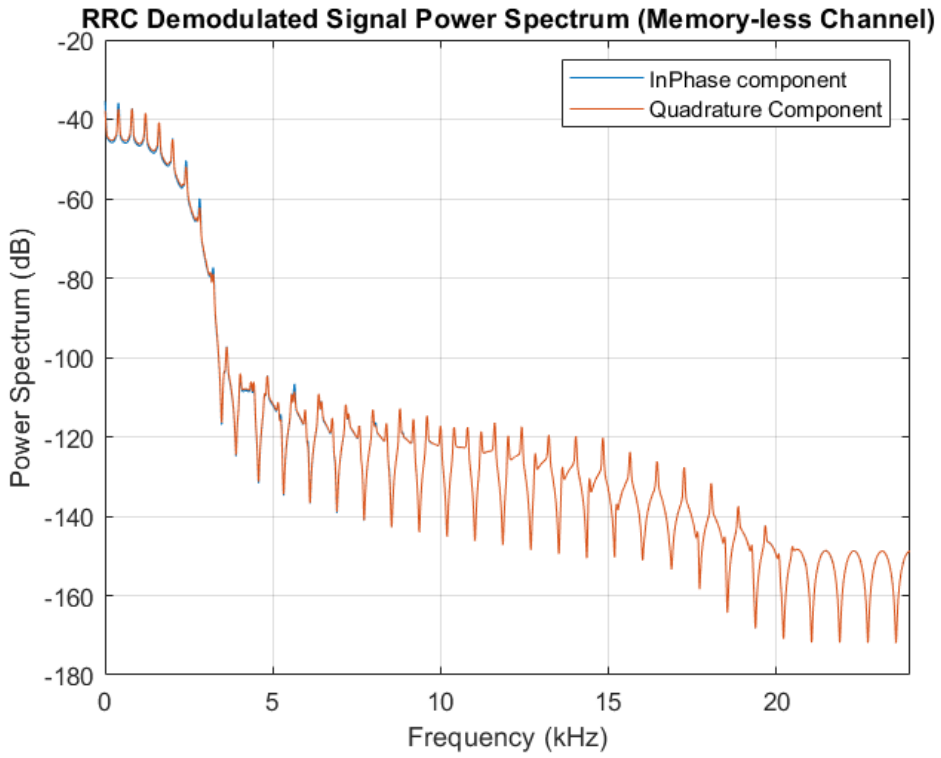

Figure 22: PSD of $y_3(t)$ when incoming signal was RC modulated and passed through memoryless channel

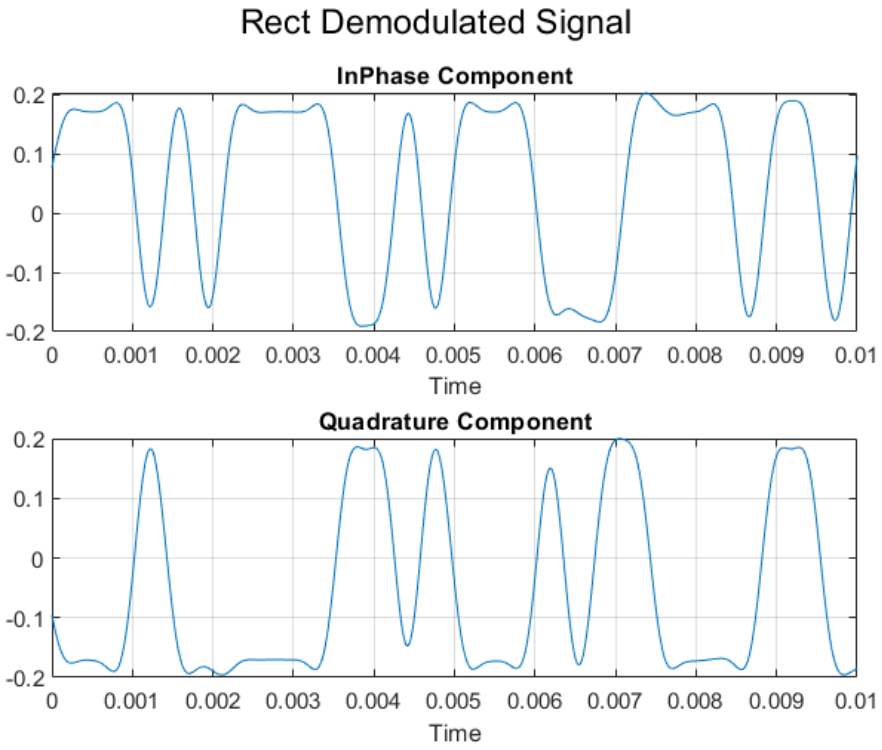

Figure 23: Rect Modulated waveform after demodulation and after memoryless channel

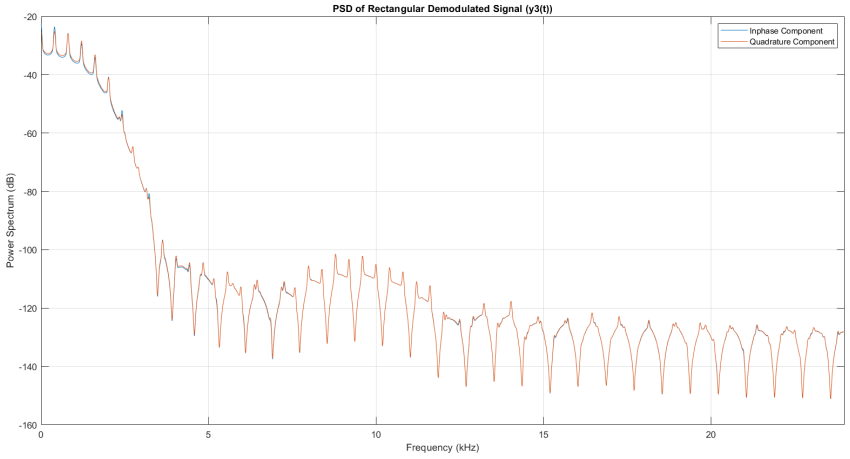

Figure 24: PSD of $y_3(t)$ when incoming signal was Rect modulated and passed through memoryless channel

8 Line Decoding

This block does the reverse process of Line Coding block i.e. it generates binary data which the decoder can further convert into bits. It takes in the waveforms of the demodulated signals and converts them into binary pulses.

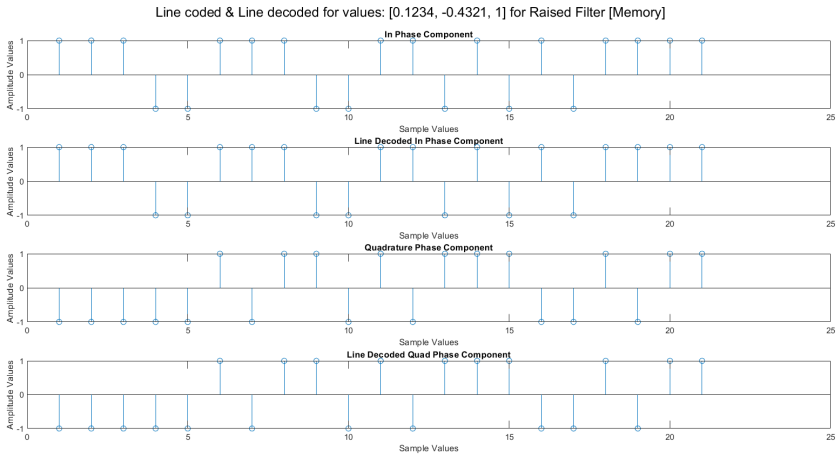

Figure 25: Line Decoded signal when received signal was RC modulated after passing through a channel with memory(a=1,b=1)

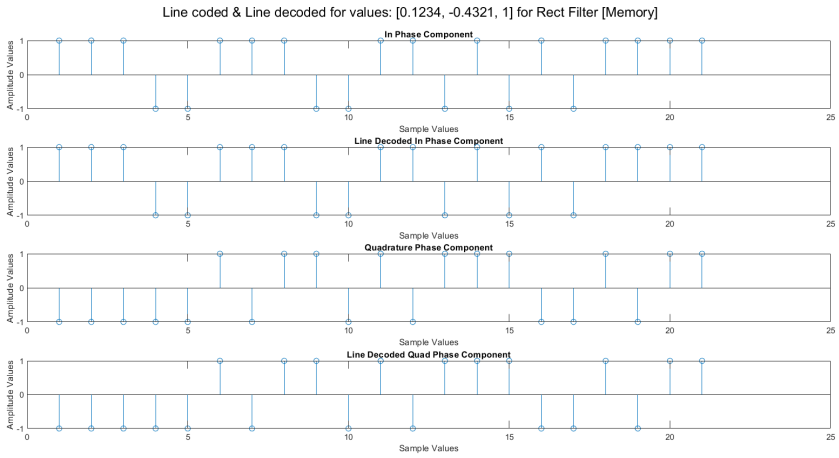

Figure 26: Line Decoded signal when received signal was Rect modulated after passing through a channel with memory(a=1,b=1)

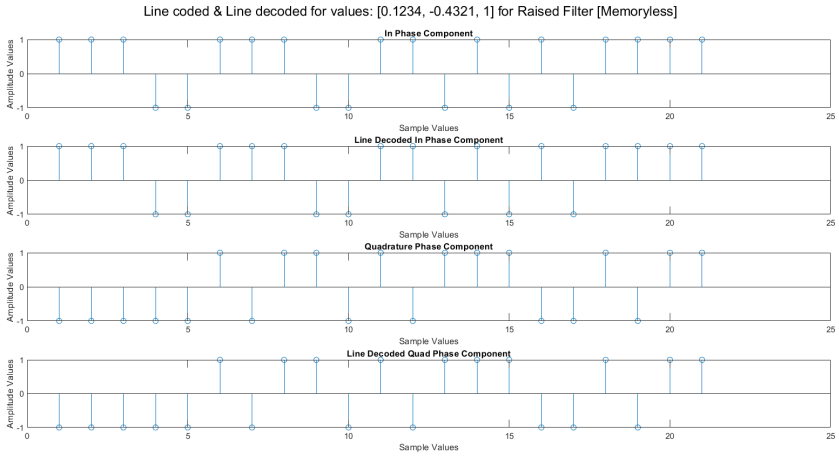

Figure 27: Line Decoded signal when received signal was RC modulated after passing through a memoryless channel

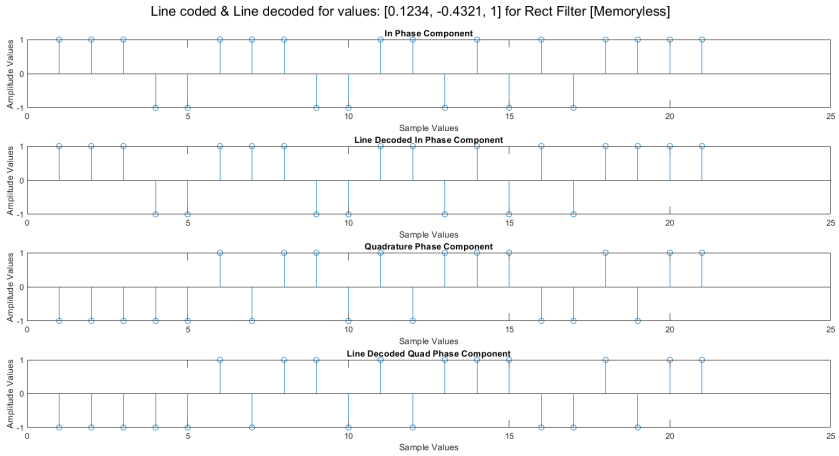

Figure 28: Line Decoded when received signal was Rect modulated after passing through a memoryless channel

9 Decoder

This block gives the final decision statistics on the basis of the inphase and quadrature signals received from the line decoder. Its mapping is reverse to that of the encoder i.e. for inphase bit = 1 and quadrature bit also = 1, it determines the decoded signal bits to be (0,0).

The constellation Mapping after decoding is given as follows for both channel realisations:

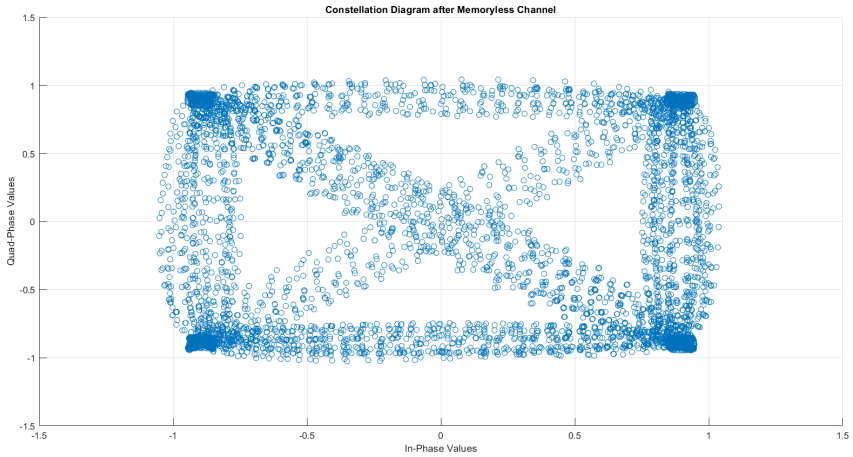

Figure 29: After Memory-less channel

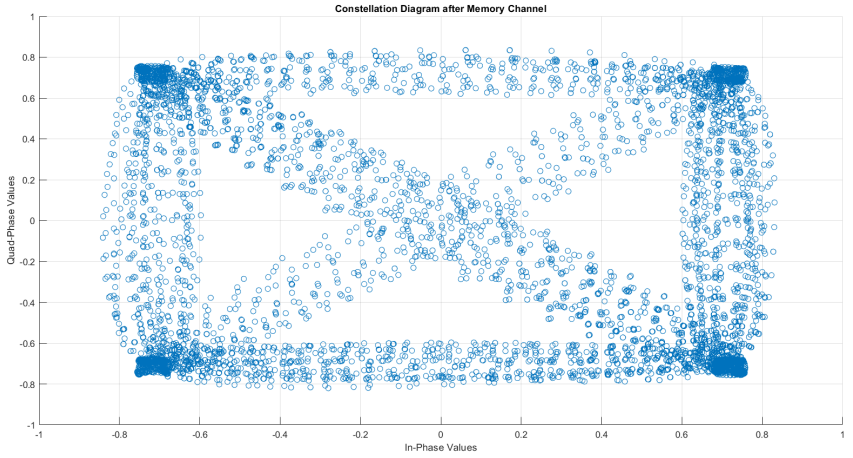

Figure 30: After Channel with Memory

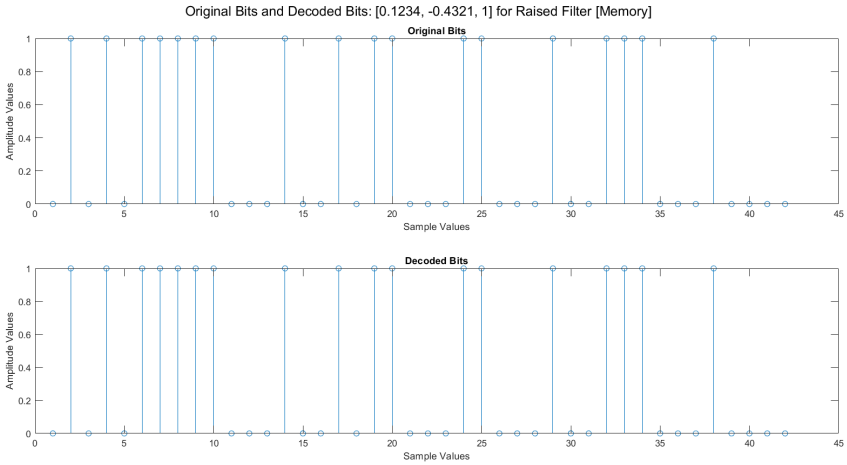

Figure 31: RC filter with memory

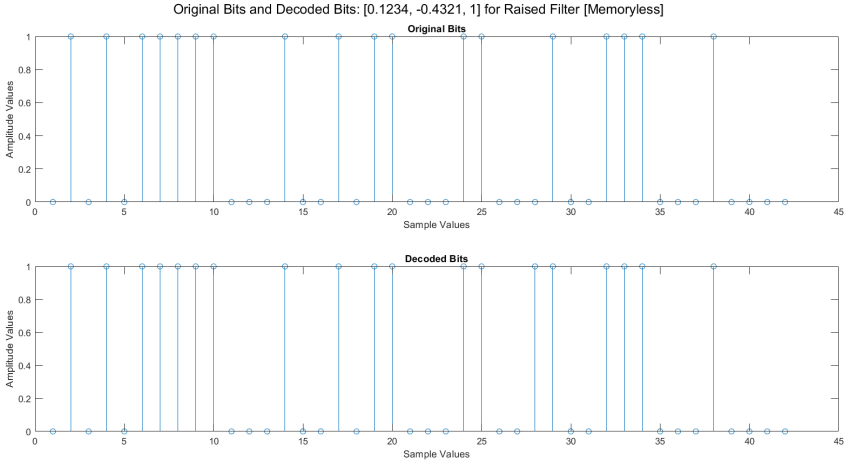

Figure 32: RC filter without memory

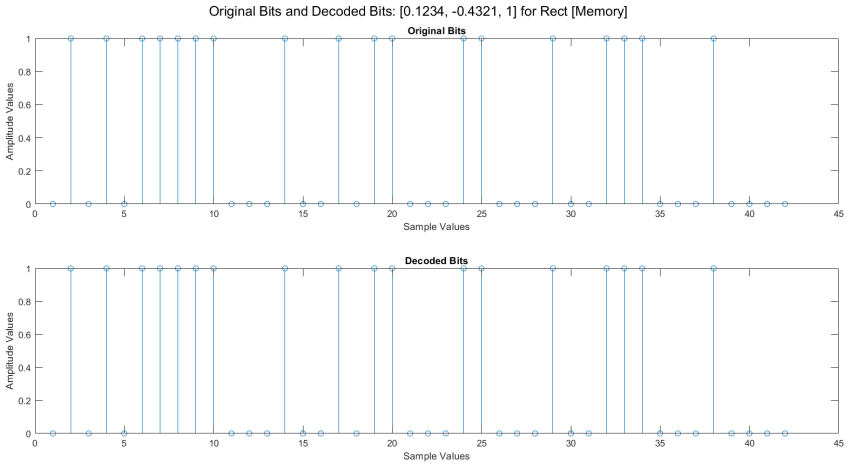

Figure 33: Rect Filter with memory

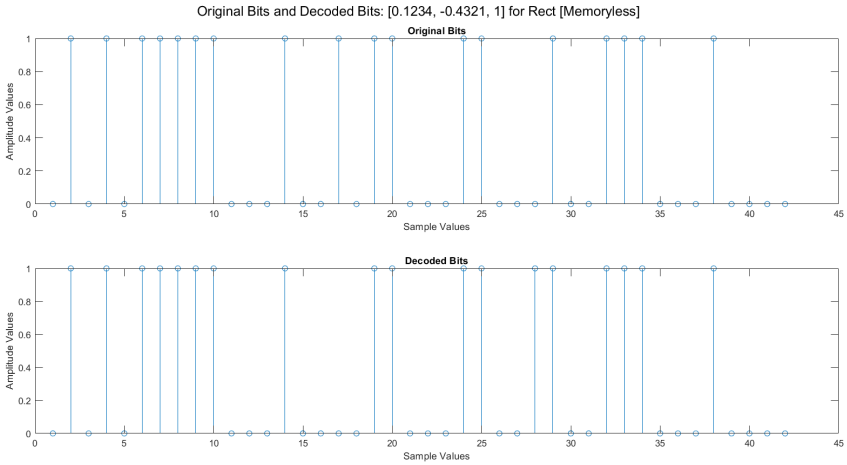

Figure 34: Rect Filter without memory

10 D2A Converter

The function d2a is designed to convert the digital signal back into an analog signal. Explanation of each step within the function is as follows:

1. Knowing that each symbol in the digital signal is represented by 14 bits, the total number of symbols is calculated by dividing the total length of the digital signal by 14.

2. The digital signal, which is a 1D array, is reshaped into a matrix where each row contains 14 bits (each symbol). The transpose (') changes the matrix orientation so that each symbol is a row.

3. 'powersOfTwo' is an array containing the powers of two from 2^{13} to 2^0, matching the bit positions in each row of binaryMatrix. Multiplying binaryMatrix by the transpose of powersOfTwo converts each binary row to a decimal value.

4. The obtained decimal values are first scaled down by dividing by 10^4 and then shifted by -0.5. This transformation standardizes the range of the values.

5. Finally, the decimal array is scaled to match the original amplitude range of the audio signal by multiplying by twice the maximum amplitude. This step converts the scaled and standardized values back into a range suitable for an analog signal, effectively creating an output that mimics the amplitude characteristics of the original mono audio.