ELEC 316 – TERM PROJECT

Ahmet Hamdi Ünal (60167)

Contents

MAY 17, 2020

KOÇ UNIVERSITY

SPRING 2020

ELEC 316 Project Report

BITRATE versus TARGET BER PLOT

I have used a 256-QAM system with raised cosines as pulse shaping filters. Since there is no extra medium, the only possible source of ISI in the system was raised cosines (carriers are far apart). Apparently, they did not cause a significant error. Therefore, I got BER of 0, except the case for $M = 2$ only. I could worsen the system, but it is not something wanted in application, and the instructor did not advise me to do so. Therefore, for different bitrates, I may present the same BER values (of 0).

Since I have implemented a 256-QAM system, $bits/symbol = \log_2 256 = 8$. Therefore, bitrate is $8/T_s$.

In the manual, it is clearly stated that carrier's sample time (which is T_S/M) must be constant and 2^{-15}. Therefore, the more we decrease M, the more T_s is decreased, which in result increases the bitrate. Therefore, when maximizing the bitrate, since system (256-QAM in this case) is fixed, M is tried to be decreased as much as possible. I should also note that M must be a positive integer because it is how many times we upsample the input signal, in other words, how many times we repeat a data point (or insert 0s, depending on the method used for upsamling). After providing this necessary information roughly, I provide the Bitrate vs Target BER plot below.

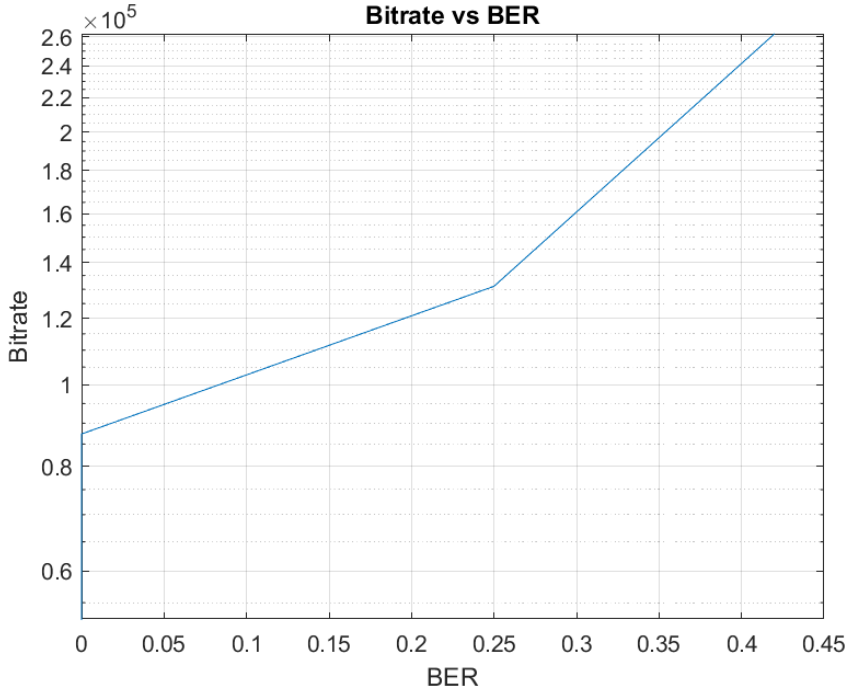


Figure 1: Bitrate vs BER

Instead of a "Bitrate vs Target BER" plot, this is a "Bitrate vs BER" plot. This is due to the fact that when $M = 2$, BER was 0.25. When I changed M to 3, BER became 0. Therefore, all the target values between 0.25 and 0 were skipped. Since carrier sample time is constant, and M can take only positive integer values, I cannot get the target values by changing M and T_s only in this setup. Therefore, we cannot observe bitrate effect for the target BERs. I could obtain target BER values by changing the parameters (other than M) of the Raised Cosine block. However, this would not change the bitrate. The instructor also did not want me to spend time on worsening the system to obtain the middle target BERs. Therefore, I have skipped them, and the result I have obtained is the figure above. Observe that up to the bitrate value of ~87381 (since the plot is plotted in *semiology*, it is not clearly visible in the figure), there is no BER. For larger bitrate values, BER exists.

TABLE OF RESULTS

Due to the reasons I have explained in the previous part, instead of Target BERs, I will fill the table below only with the BERs I was able to obtain. The range of BERs obtained is $\{0, 0.25, 0.42\}$, which can also be observed in Figure 1. 0.25 BER belongs to the case $M = 2$ and 0.42 BER belongs to the case $M = 1$, which is sending bits directly (no sampling). When $M > 2$, BER is always 0.

BER	M	Symbol rate	Const. Type	Const. Size	Min. Distance Between Symbols	Carrier Sample Frequency (Hz)	Raised Cosine Rolloff Factor	Raised Cosine Filter Span	Raised Cosine Output Samples /Symbol	Down-sample Factor
0.42	1	2^{15}	256 QAM	15	2	2^{-15}	–	–	–	–
0.25	2	2^{14}	256 QAM	15	2	2^{-15}	0.2	10	2	2
0	>2	$2^{15}/M$	256 QAM	15	2	2^{-15}	0.2	10	M	M

Table 1: BER vs System Parameters

Carrier frequency is 10 kHz, and it will always be 10kHz throughout this report. Simulations stop time is also set for 60 throughout this part (main part) of the project.

1. BER = 0.42

a. Design Approach Description

M is set to the lowest possible value, which is 1. M being equal to 1 means that there is no up-down sampling. Therefore, QAM modulator block output bits are sent directly without any pulse shaping filter. For this case, I have removed Raised Cosine blocks and the Downsample block. Since $M = 1$, symbol rate is 2^{15}. This is the case we expect the highest bitrate, but due to ISI (δ is infinite in bandwidth), the output would probably give out meaningless values with a high BER.

b. Simulink Diagram Screenshot with Performance Results

I provide the Simulink model for this setup below. It is very similar to the setup used for Experiment 7. The Tx, Rx Modules and ErrorRate Calculation block is updated, modulator and demodulator blocks are changed to 256 QAM and since here I investigate the case M=1, upsampling – downsampling blocks are removed. The rest is as usual: Modulator maps the signal into real and complex values, which are transmitted with sine and cosine block separately.

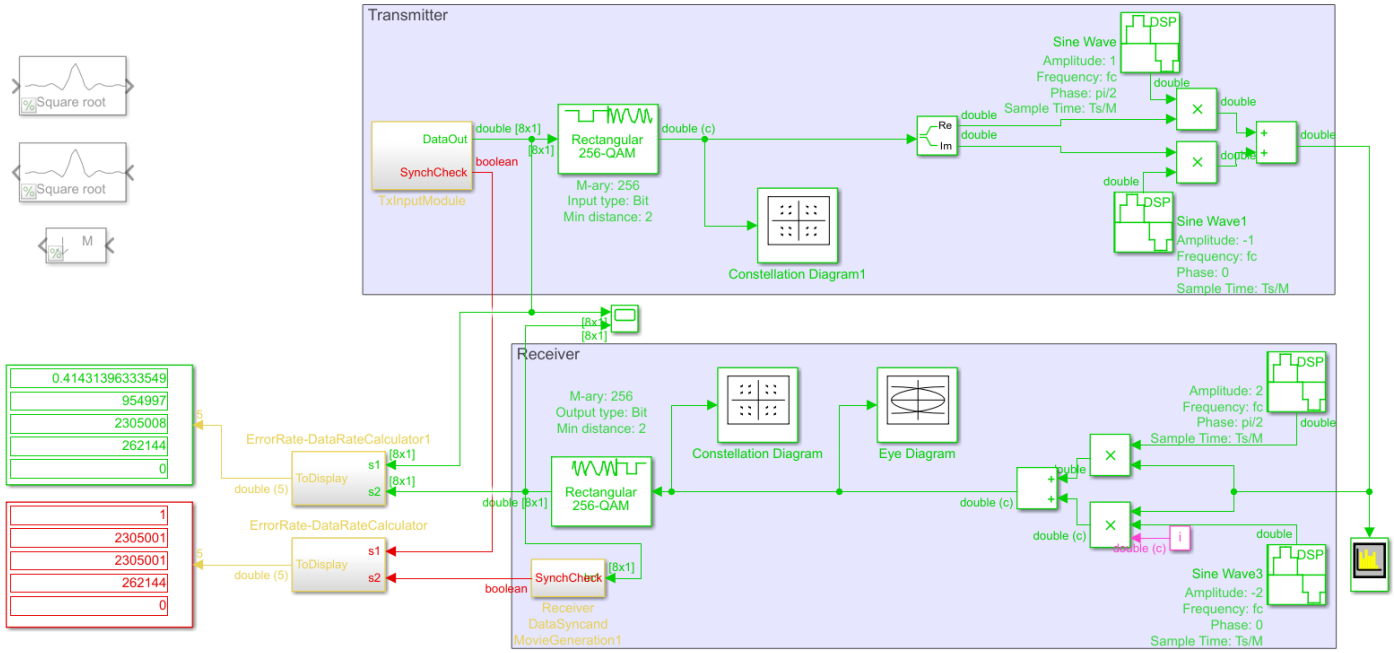


Figure 2: Simulink model for BER = 0.42

BER=0.42, Bit Rate= 262144

Stop time is set to 60. BER is calculated by using 8/Ts.

c. Simulation Parameters Table for the current Target BER

BER	M	Symbol rate	Const. Type	Const. Size	Min. Distance Between Symbols	Carrier Sample Frequency (Hz)	Raised Cosine Rolloff Factor	Raised Cosine Filter Span	Raised Cosine Output Samples /Symbol	Down-sample Factor
0.42	1	2^{15}	256 QAM	15	2	2^{-15}	–	–	–	–

Table 2: Table 1 (cropped) for BER of 0.42

Detailed parameter block screenshots are to be provided in the upcoming parts of this report.

d. Transmit Constellation Plot

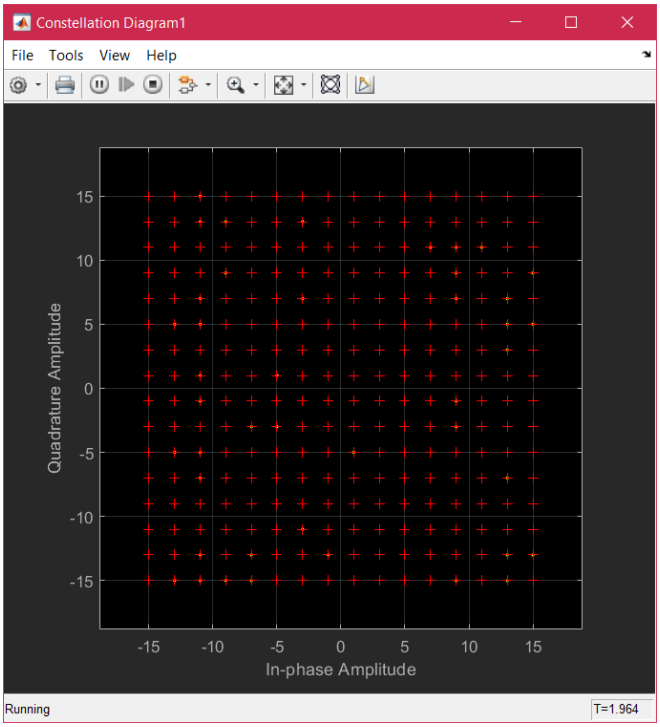


Figure 3: Transmit Constellation Plot for BER of 0.42

It may not be visible, but there are yellow points in the figure that fall just onto the red crosses. This is as expected because signal is not subjected to any external effect yet. This is the case just after mapping is performed.

e. Received Constellation Plot (A Representative Example)

Below, I provide a screenshot of Received Constellation Plot during run. Since there is empty space here, I will write the points that I want to emphasize about this plot here. I have noted that single data points are represented as Kronecker-Delta in the discrete time. The Fourier Transform of a Kronecker-Delta is $\vec{1}$, which extends to infinity. It is not a damping oscillation like raised cosine or sinc though. It is a constant one, which guarantees that it is going to mess up the frequency domain. This is what happens in this case too. The dispersed constellation points below can be explained with this. Since spectrum is not required, I did not provide it here, but it is not difficult to guess what happens. $\vec{1}$ convolves with sine-cosine in the frequency domain, which in result gives out undistinguishable peaks along with a useless spectrum.

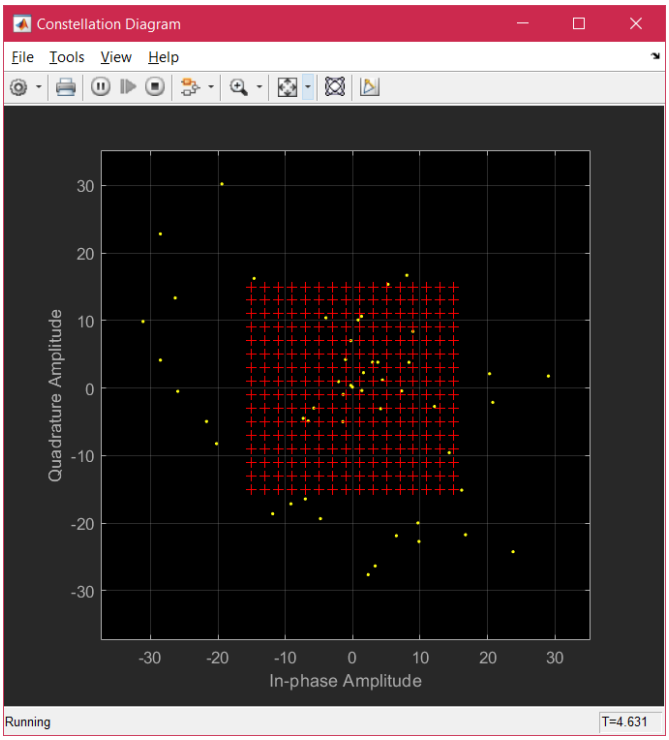


Figure 4: Received Constellation Plot for BER of 0.42

f. Eye Diagram Plot (at the input of demodulator)

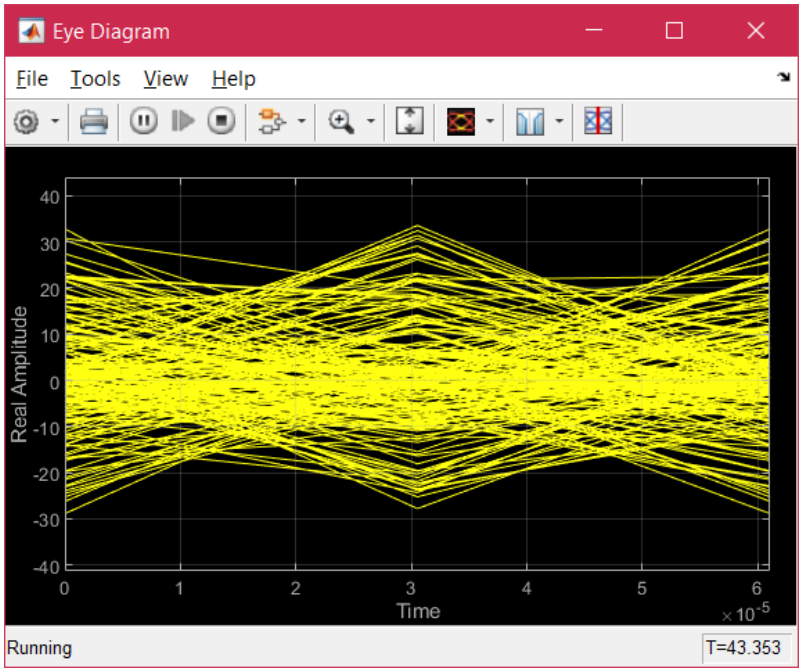


Figure 5: Eye Diagram plot for BER of 0.42

The eye is completely closed. Therefore, it is not possible to sample at Ts and get back the signal accurate enough. This eye diagram agrees the constellation diagram, which indicates that the receiver is hopeless.

g. Screen shots for Block Parameters

The required parameters of the blocks are visible in the screenshot of the system I have provided. Nevertheless:

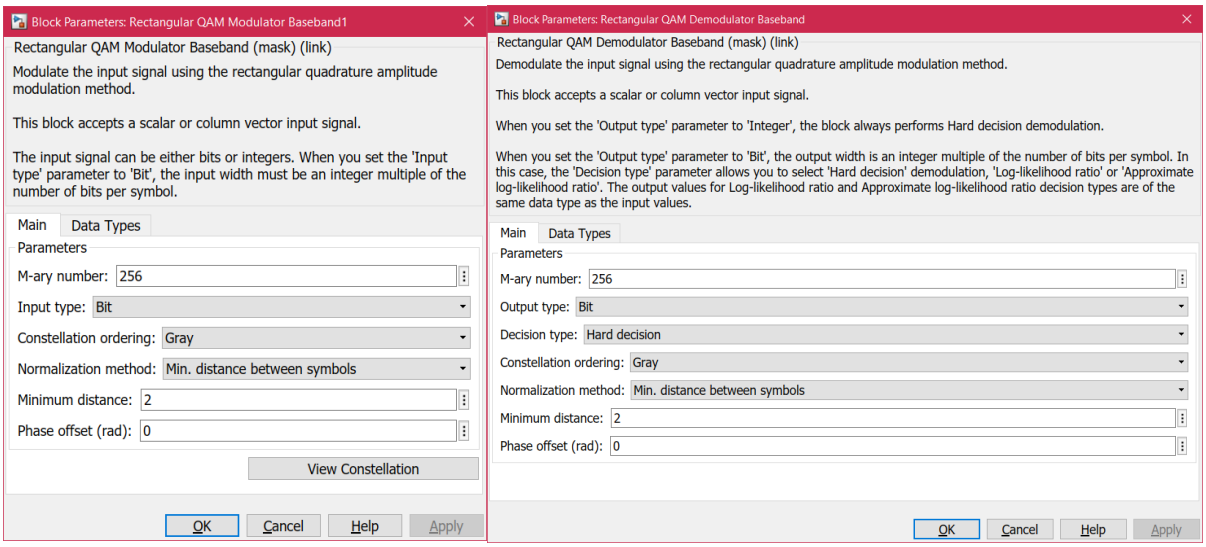


Figure 6: Parameters of QAM Modulator – Demodulator Blocks

M-ary denotes the Constellation Size. Input/Output type is chosen as *bit* since both Tx, Rx blocks work with bits. Minimum distance is set to 2. This is mostly enough and what we have always covered in the lectures. Increasing it requires increase in power, which can be done actually in this case. However, when I tried to increase it, I did not observe a significant change in the BER. So, I have decided to use the conventional size. Increasing it would decrease BER, but in this case, it cannot recover the effect of a BER of 0.4.

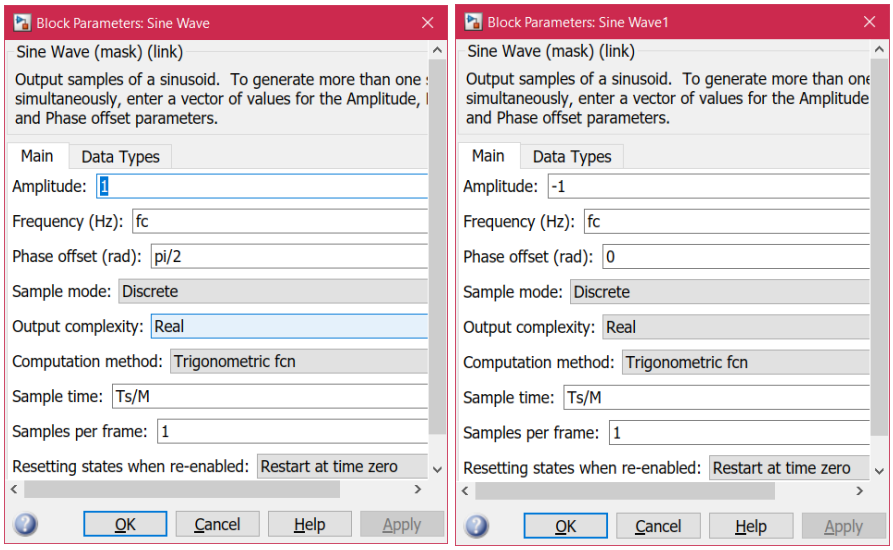


Figure 7: Transmitter Carrier Parameters

I set carrier frequencies to 10kHz so that interference would probably not be a problem in any case (except infinite bandwidths). As stated, carrier sample time is Ts/M.

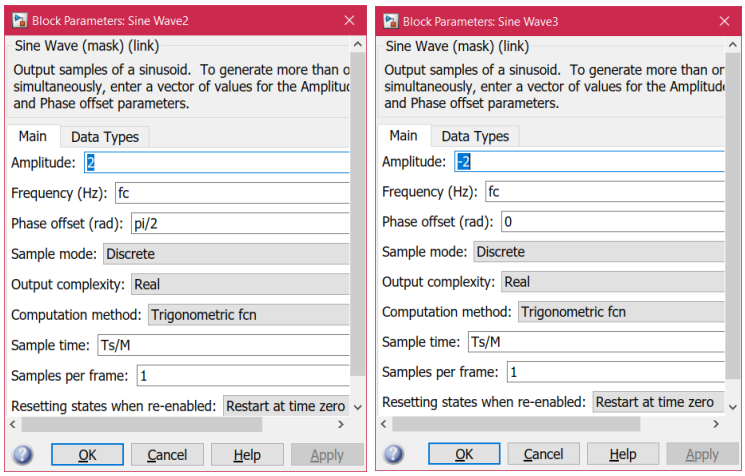


Figure 8: Receiver Carrier Parameters

Carriers' sample times are set to T_s/M, which is always equal to 2^{-15} in this experiment. f_c is set to 10 kHz to avoid frequency interference in the frequency domain, which is not likely in this case. In short, similar settings are applied to the carriers at both sides. Minor differences are due to transmission method, which is already covered in detail in Experiment 7.

h. Discussion of Results

In this scenario, we have considered the case where there is no pulse shaping and up sampling in the transmitter. The output bits of the 256-QAM Modulator is sent directly to the receiver after being modulated with the carriers. This creates trouble. When pulse shaping process is removed, each bit becomes a single hit, which we can model as a δ. the problem with δ pulse in time domain is that its not being limited in the frequency domain ($F\{\delta[n]\} = 1$). Therefore, it is unavoidable that the frequencies will mix up, and this interference is unignorable since 1 is constant for all frequencies. Therefore, such poor result was expected. The more we upsample and give pulse a shape (preferably raised cosine), the more the signal will be frequency-limited, which will in return decrease ISI. Without looking at BER, Constellation and Eye Diagrams tell us how bad this transmission is. Points are floating around in the Constellation Diagram (see Figure 4) and eye is completely indistinguishable (see Figure 5). Therefore, though this is the highest bitrate case, this bitrate is meaningless if we get almost half of the bits wrong in the end.

2. BER = 0.25

a. Design Approach Description

This time, pulse shaping blocks are inserted. Since raised cosine is superior to rectangle shaped pulses (because of its oscillations are being controllable and ignorable up to a rate, whereas rectangular pulses are unlimited in the frequency domain), I prefer raised cosines pulses. In this scenario, upsampling (M) is 2. For this purpose, a Raised Cosine Transmit Filter after QAM modulator block is inserted, along with a Raised Cosine Receive Filter before QAM demodulator block. Also, I have placed a Downsample block just after this Raised Cosine Receive Filter, so that the bit sequence can be extracted from the raised cosine pulses received. Note that both raised cosine filters' shapes are chosen as Square Root, so no Gain block is required in the end.

b. Simulink Diagram Screenshot with Performance Results

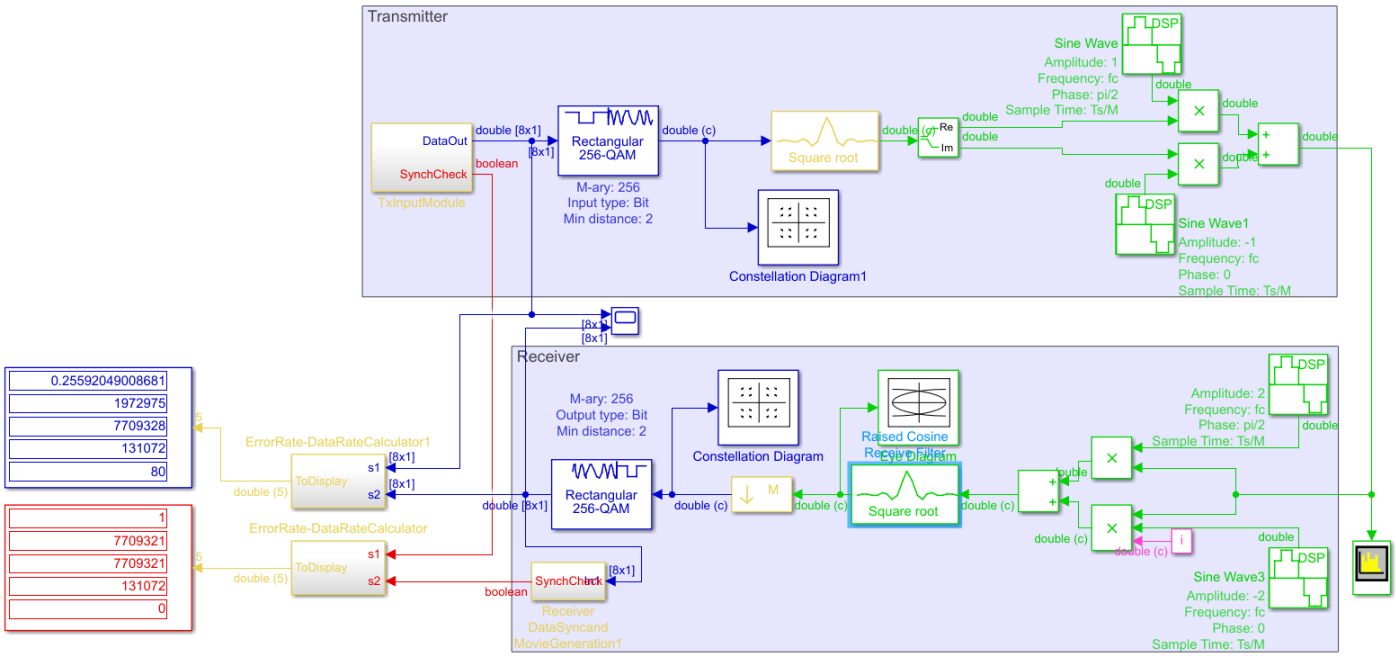


Figure 9: Simulink model for BER = 0.25

BER=0.25, Bit Rate= 131072 bits/s

Stop time is set to 60.

Different from the previous setup, this setup includes pulse shaping (upsampling at the same time) blocks, along with down a sampling block. I believe that in the previous section, I have explained why we need pulse shaping and upsamling clear enough. The rest of the setup is the same.

c. Simulation Parameters Table for the current Target BER

BER	M	Symbol rate	Const. Type	Const. Size	Min. Distance Between Symbols	Carrier Sample Frequency (Hz)	Raised Cosine Rolloff Factor	Raised Cosine Filter Span	Raised Cosine Output Samples /Symbol	Down-sample Factor
0.25	2	2^{14}	256 QAM	15	2	2^{-15}	0.2	10	2	2

Table 3: Table 1 (cropped) for BER of 0.25

d. Transmit Constellation Plot

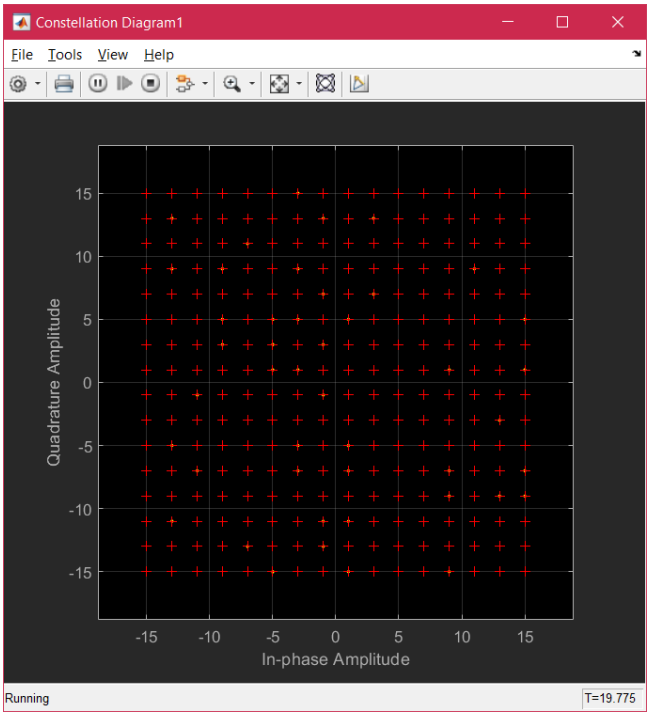


Figure 10: Transmit Constellation Plot for BER of 0.25

It may not be visible, but there are yellow points in the figure that fall just onto the red crosses. This is as expected because signal is not subjected to any external effect yet. This is the case just after mapping is performed.

e. Received Constellation Plot (A Representative Example)

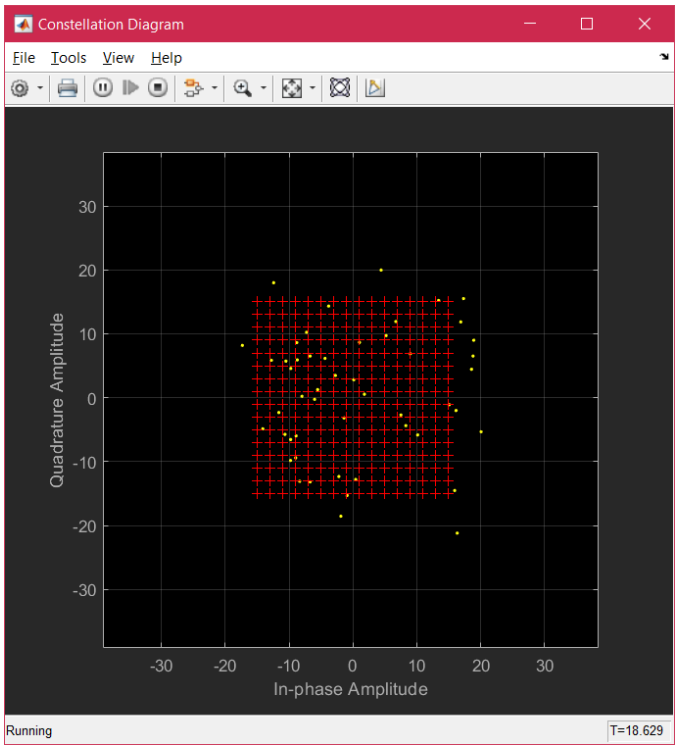


Figure 11: Received Constellation Plot for BER of 0.25

As the constellation diagram in the figure below indicates, it is not a good idea to expect a satisfactory result for this case. Constellation points are again floating away. This time we have inserted pulse shaping blocks and upsampled though. The reason why the constellation is this much scattered will be investigated in the end. At this point, it is enough to note that constellation points are far from where they should be, so ISI occurred that we cannot distinguish the bits anymore.

f. Eye Diagram Plot (at the input of demodulator)

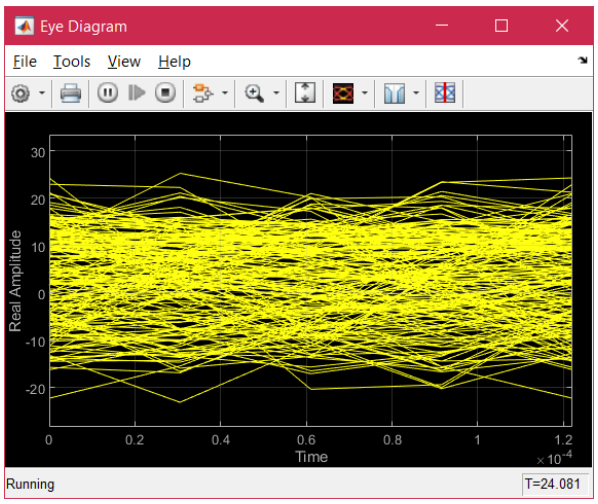


Figure 12: Eye Diagram plot for BER of 0.25

The eye diagram once again verifies the constellation diagram. The eye is completely closed. Therefore, it is not possible to sample at Ts and get back the signal accurate enough. This eye diagram agrees the constellation diagram, which indicates that the receiver is hopeless.

g. Screen shots for Block Parameters

256-QAM Modulator – Demodulator and Carrier Blocks are not changed. So, their properties are the same. See Figure 6, Figure 7 and Figure 8 for their properties. In addition, Raised Cosine Blocks are added. Their properties are as follows:

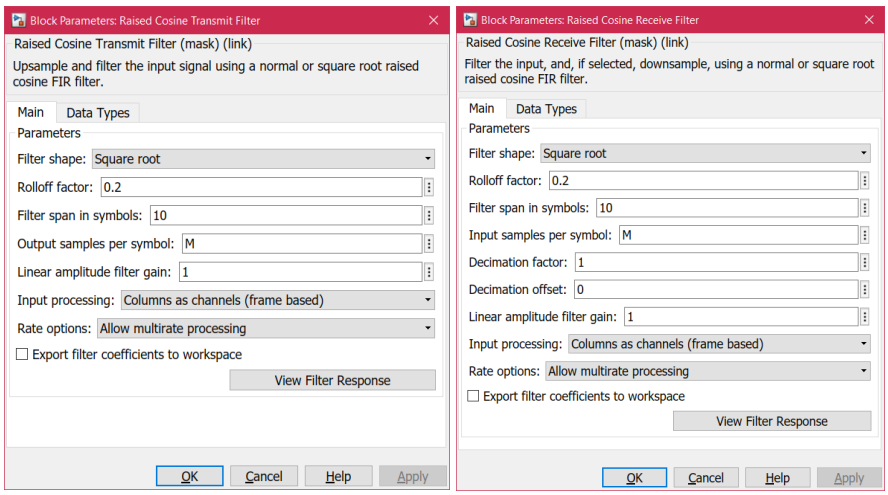


Figure 13: Raised Cosine Filters Properties

Rolloff factor is 0.2, meaning that this raised cosine is close to sinc shapewise. It decays faster than the sinc, but not very much faster as roll of factor suggests. Filter span is set to 10, meaning that in 10 symbols, raised cosine completely disappears. This is not realistic since it extends to infinity. However, because this is a raised cosine and not a sinc, these extensions can be ignored. So, 10 is acceptable. Input samples per symbol is set to M, which means that this block upsamples the signal by a factor of M (2 in this case).

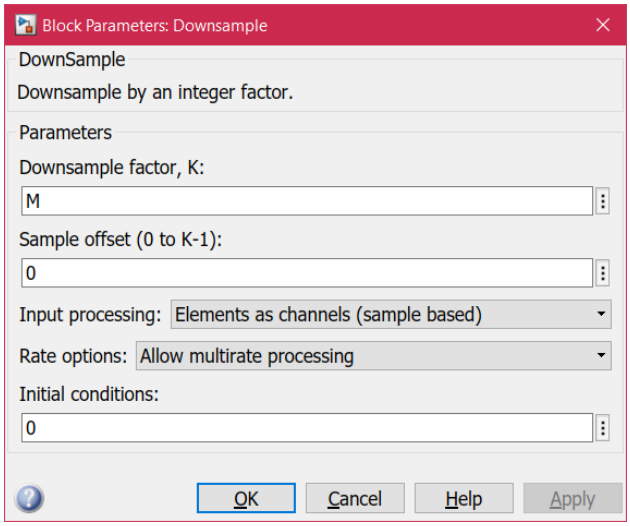


Figure 14: Downsample Block Properties

This block simply downsamples its input, meaning that with this block, the peak values of each raised cosine received is extracted, if down sampled at the right Ts and starting at the right time to downsampling. These are our QAM modulated bits, so they must be de-mapped in the QAM demodulator block, which is right after this block.

h. Discussion of Results

BER of 0.25 is not a satisfactory result. Nonetheless, it is expected in this case. When compared to the previous case, BER is decreased. The more M is increased, we expect a better representation of a raised cosine. In the previous case, there was no sampling and result was extremely poor. In this case, raised cosine pulses are to be constructed but raised cosines consisted of 2 data points for each bit is also a poor case. When $M \to \infty$, it is sure that the best raised cosines are to be got. However, that much points are unnecessary for a good approximate representation of a raised cosine. Here, we have observed $M = 2$ was not enough, so we require a greater upsampling in the pulse shaping case. If we do not get a good raised cosine, we cannot control the behavior of the signal in the frequency domain, just as the case in the previous case with BER of 0.42. This upsampling amount discussion here is the same as in the last experiment. We have asserted that we are to work with raised cosines, but if we represent the raised cosine with, say, 1 point, you are not considered to work with raised cosines. So, for pulse shaping, M is necessary and must be increased to the point we get an acceptable result. To get a more frequency-limited signal, here we must increase M. Constellation and Eye Diagrams in Figure 10 and Figure 11 verifies this. Next, I will increase M to 3, and check whether it gives a satisfactory result. If it does not, it means that M should be increased more.

3. BER = 0.

a. Design Approach Description

The only change I do in this part is to set $M = 3$, which in returns changes T_s to $Ts = (2^{-15}) * 3$. Examining this case for $BER = 0$ is enough because the more we upsample, the better raised cosines we get so that results will only get better, not worse. Also, if $BER = 0$ is got in this case, this becomes the highest bitrate case that gives no error, so increasing M after this value would be pointless (if we get BER=0 of course, which is the case as the heading suggests).

b. Simulink Diagram Screenshot with Performance Results

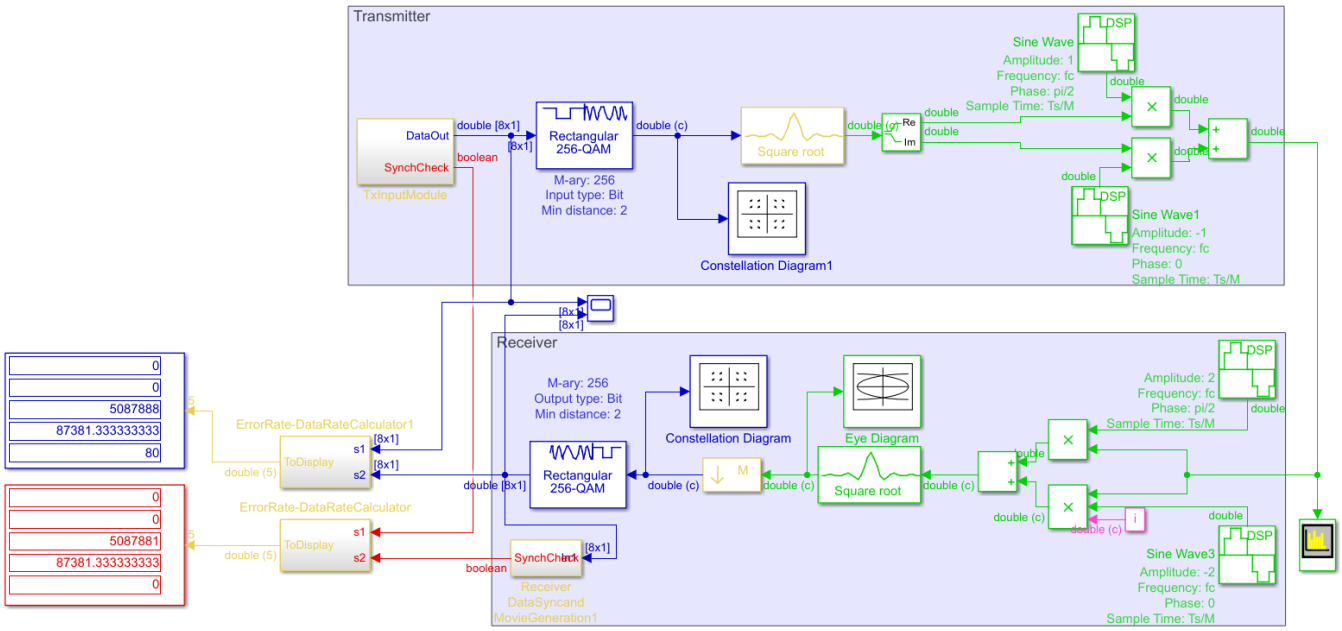


Figure 15: Simulink model for BER = 0

BER=0, Bit Rate= ~87381 bits/s

Stop time is set to 60.

I will not provide an extra explanation about this setup since it is the same as in the previous case. All changes (M and Ts) are done on MATLAB command windows using coding skills.

c. Simulation Parameters Table for the current Target BER

BER	M	Symbol rate	Const. Type	Const. Size	Min. Distance Between Symbols	Carrier Sample Frequency (Hz)	Raised Cosine Rolloff Factor	Raised Cosine Filter Span	Raised Cosine Output Samples /Symbol	Down-sample Factor
0	3	$2^{15}/3$	256 QAM	15	2	2^{-15}	0.2	10	3	3

Table 4: Table 1 (cropped and modified) for BER of 0 when M=3

d. Transmit Constellation Plot

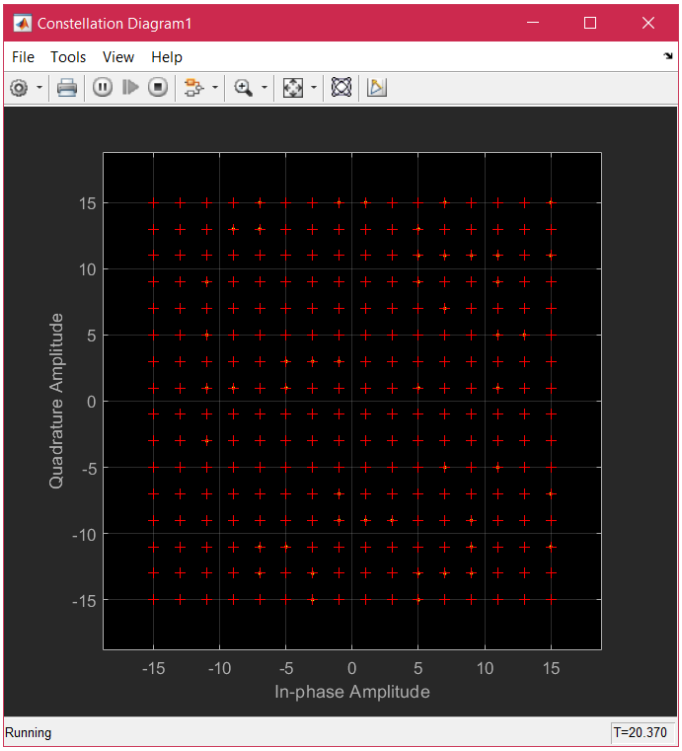


Figure 16: Transmit Constellation Plot for BER of 0

It may not be visible, but there are yellow points in the figure that fall just onto the red crosses. This is as expected because signal is not subjected to any external effect yet. This is the case just after mapping is performed.

e. Received Constellation Plot (A Representative Example)

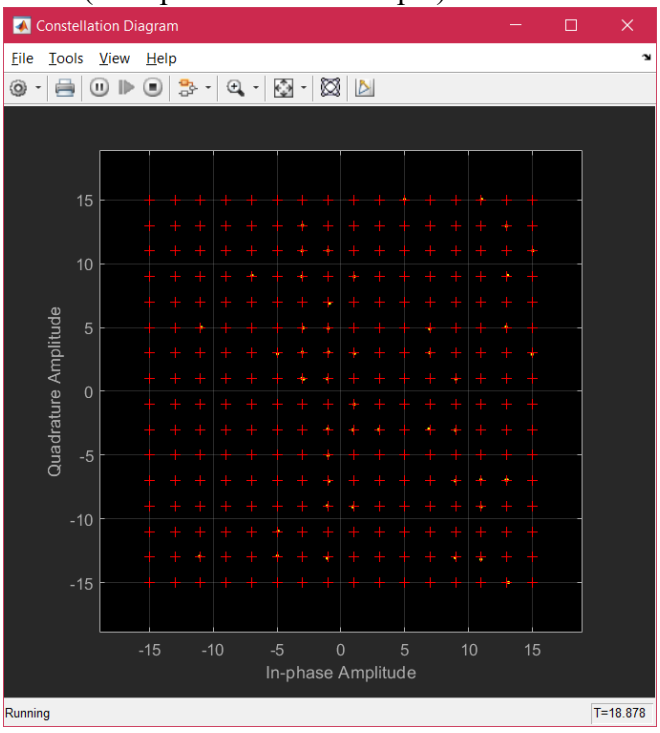
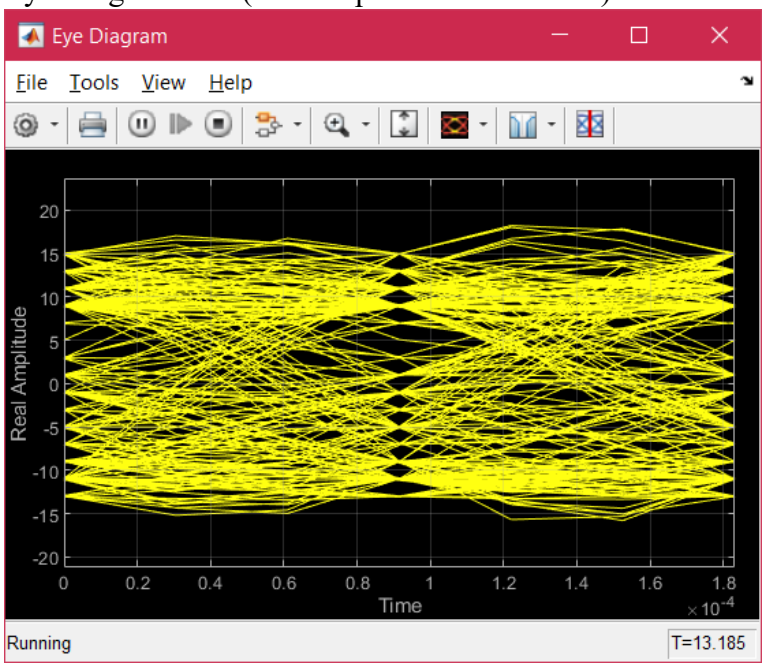


Figure 17: Received Constellation Plot for BER of 0

It may not be visible, but there are yellow points in the figure that fall just onto the red crosses. Finally, constellation points are on the points where they should be. It means that the transmission was successful and ISI, if occurred, did not caused much harm on the bits, so that they still fall around the designated points.

f. Eye Diagram Plot (at the input of demodulator)



Eye at the sampling time ($Ts = (2^{-15}) \times 3 \approx 9x10^{-5}$)) is fully open. Therefore, we can easily recover the original signal back. Both constellation diagram and eye diagram are encouraging. See the holes in the middle of the eye diagram: they are fully open.

g. Screen shots for Block Parameters

Block parameters are exactly the same. Not a single thing is changed in the model. The only change in the system is M and Ts, which are changed through MATLAB command window without touching the Simulink model. See the Block Parameters section of the case when BER=0.25.

h. Discussion of Results

In this part, for M=3, we have successfully recovered the signal back, with a BER of 0. It could not be better. So, we conclude that for this setup, M=3 was enough. At the beginning of the report, I have noted that since there is no medium (virtual or physical) inserted during transmission, and carrier frequencies are much far apart, the only possible cause of ISI would be raised cosines (or infinite BW signals as in the case with M=1). Here, I have observed that ISI caused by raised cosines is not a problem, it is ignorable. M=3 is a very small value, in fact, it is probably the lowest value (M=2 is not realistic due to the explanations I have provided here.) Even so, there is no error. Therefore, raised cosines are a very good pulse shaping tools that works. For comparison, when I run this setup with rectangular pulses, system does not work at M=3. Also, once again, it is verified how informative constellation and eye diagrams are.

BONUS PART: ADDING AWGN BLOCK

For this part, I run simulations for the stop time: 10. The setup is the same, so I do not provide a screenshot. The only inserted block is AWGN, which is connected to the output of the transmitter. The AWGN block I use has the following properties:

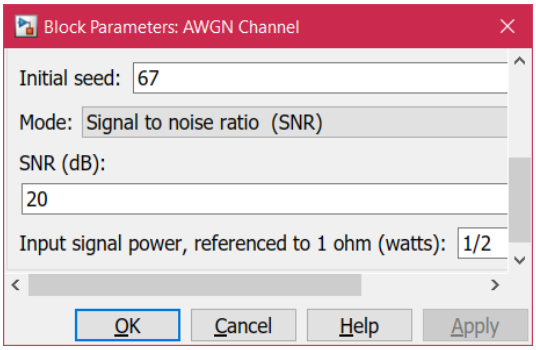


Figure 18: AWGN Block Properties

The input signal power is set to ½ because Linear amplitude filter gain of the Raised Cosine Transmit Filter is 1 (no effect), and there is sine and cosine multiplications, which in result affects the input power as $(1/2)^2 + (1/2)^2 = 1/2$.

SNR: 20

This case is similar to the ones that no channel is inserted. As in the previous parts, system worked with 0 BER for the values $M > 2$. For $M = 2$, synchronization is failed, and BER is the same as in the case when there was no AWGN block (BER of 0.25). Therefore, for this SNR value, I only can note the following properties:

- Maximum bitrate: $M = 3 \rightarrow 8/Ts = 8/((2^{-15}) \times 3) = 87381$ for $BER = 0$
- Bitrate: $8/((2^{-15}) \times 2) = 131072$ for $BER = 0.25$.

SNR: 15

This case is also a particularly good one.

- For M=4 ($Ts = (2^{-15}) \times M$), BER is 0.
- For M=3 ($Ts = (2^{-15}) \times M$), BER is 10^{-6}. I consider this as best bitrate case since BER is smaller than 10^{-5} target BER value. Therefore, for this SNR, maximum bitrate is -again- 87381 bits/s.
- For M=2, the system fails and gives a BER of 0.25 as in the previous part.

SNR: 10

- For M=8, BER is 3(10^-4)
- For M=7, BER is 4(10^-4)
- For M=6, BER is 4(10^-4)
- For M=5, BER is 4(10^-4)
- For M=4, BER is 4(10^-4)
- For M=3, BER is 4(10^-4)
- For M=2, system fails with the same BER (0.25)

For higher M values (lower bitrates) I could not get a better BER. I have tried up to M=2^8 (bitrate = 1024). For higher M values, it had to set stop time more than 200, so I stopped trying. Therefore, higher bitrate achieved is 87381 bits/s.

SNR: 5

- For M=5, BER is 10^-2
- For M=4, BER is 10^-2
- For M=3, BER is 10^-2
- For M=2, BER is 0.26

For higher M values (lower bitrates) I could not get a better BER. I have tried up to M=2^8 (bitrate = 1024). For higher M values, it had to set stop time more than 200, so I stopped trying. Therefore, higher bitrate achieved is 87381 bits/s.

SNR: 0

- For M=3, BER is 7(10^-2)
- For M=2, BER is 0.26

For higher M values (lower bitrates) I could not get a better BER. I have tried up to M=2^8 (bitrate = 1024). For higher M values, it had to set stop time more than 200, so I stopped trying. Therefore, higher bitrate achieved is 87381 bits/s.

SNR: -5

- For M=3, BER is 10^-1
- For M=2, BER is 0.27

For higher M values (lower bitrates) I could not get a better BER. I have tried up to M=2^8 (bitrate = 1024). For higher M values, it had to set stop time more than 200, so I stopped trying. Therefore, higher bitrate achieved is 87381 bits/s.

Unfortunately, highest bitrate achieved is turned out to be the same for all SNR values. However, depending on the SNR value, corresponding BER values change, which is what we expect.

For each target BER (and other BER values I was able to obtain (other than 0.26-7)), best bitrate is 87381 bits/s. This is independent of the SNR value. Of course, this is true up to some rate. For example, there has been minor changes in the BER values for the same BER but different bitrates. Since these changes were not in the order of 10^-1, I did not insert them in here.

Below, I provide my plots. Since the range of BERs I was able to get is limited, I will plot the graphs with all the BERs I was able to get. After providing plots, I will make an explanation, and then provide the table.

PLOTS:

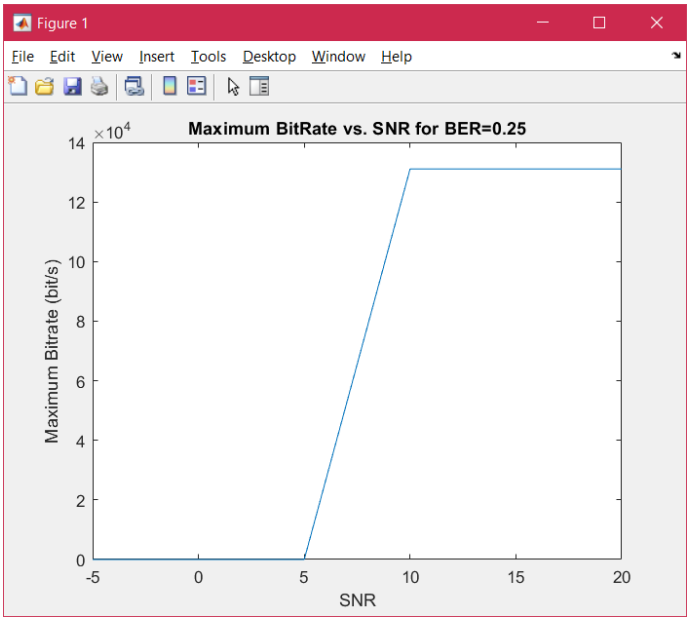


Figure 19: Maximum BitRate vs SNR for BER=0.25

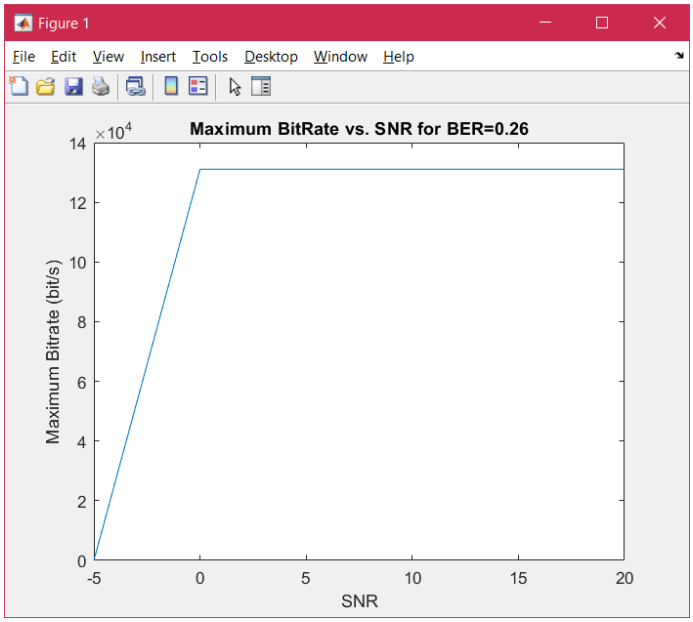


Figure 20: Maximum BitRate vs SNR for BER=0.26

From Figure 16-17, it is observed that maximum bit rate increases with SNR for a given BER.

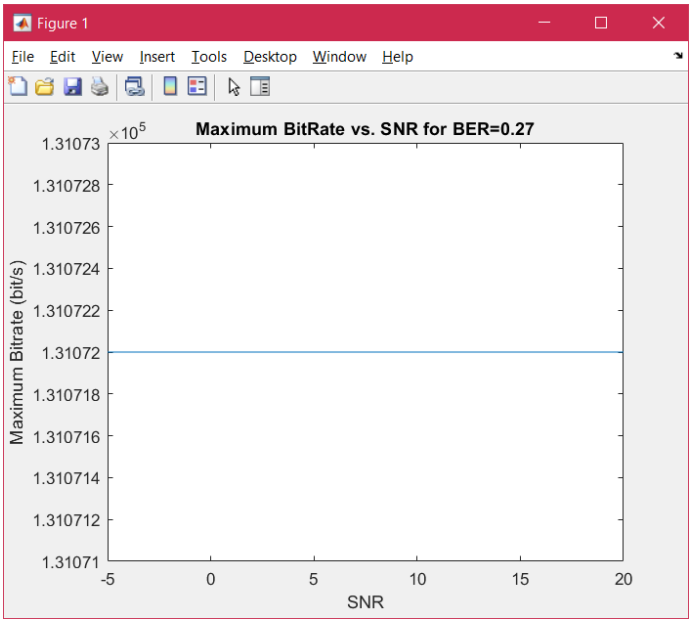


Figure 21: Maximum BitRate vs SNR for BER=0.27

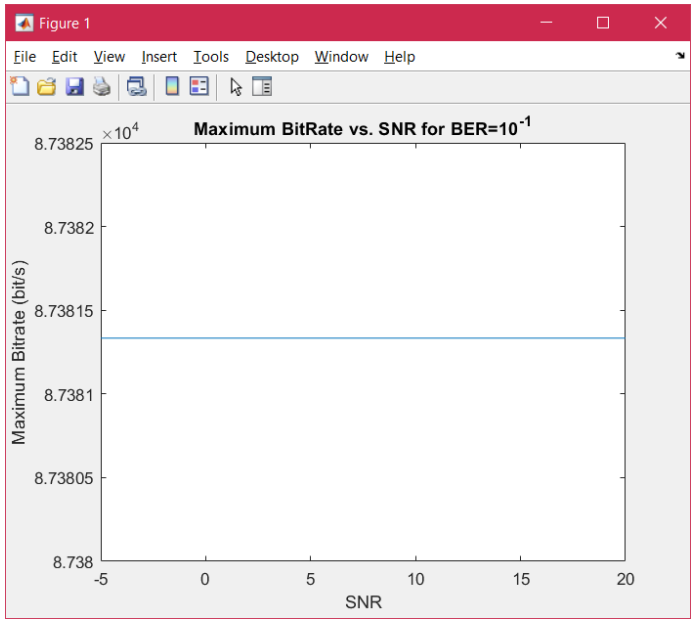


Figure 22: Maximum BitRate vs SNR for BER=0.1

From Figure 18-19, it looks like maximum BER is constant for all SNR values. However, this is not the case. Both maximum bitrate values are obtained for SNR=-5 only. So, each plot shows the correct value at SNR=-5. For the rest, I did not insert 0 because then it would look like maximum bitrate decreases with increasing SNR (which is definitely not correct). Therefore, I have made the plot constant, indicating the minimum value for higher SNRs.

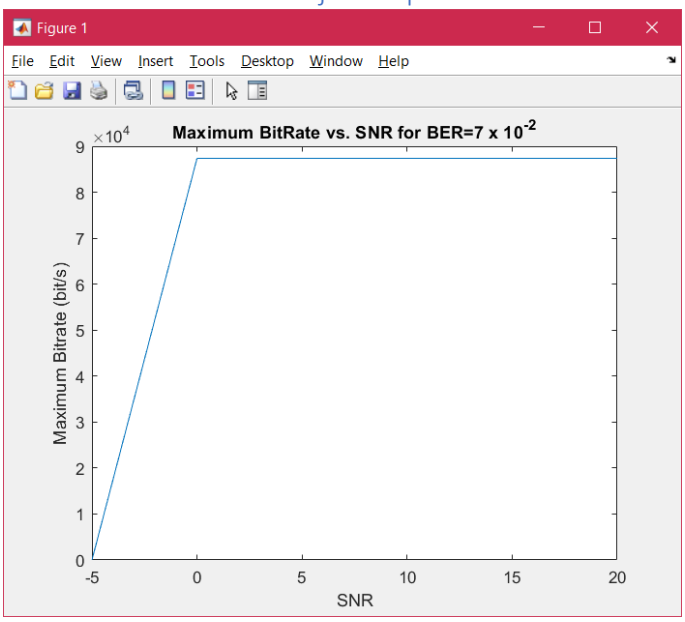


Figure 23: Maximum BitRate vs SNR for BER=0.07

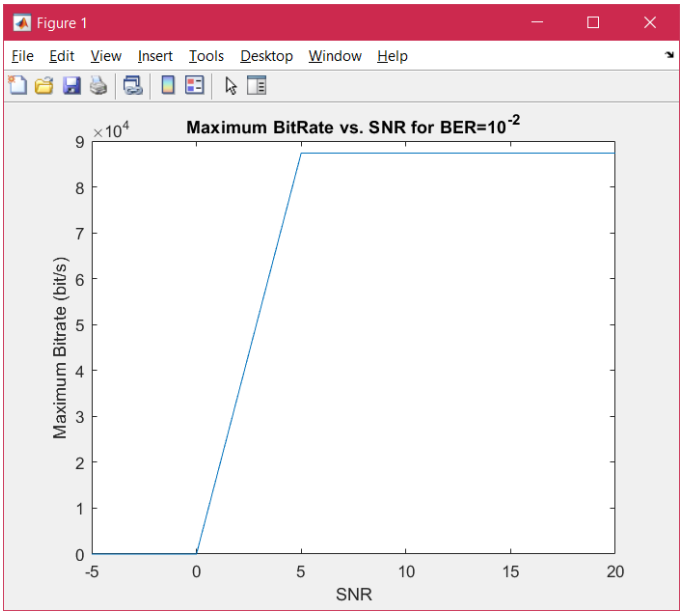


Figure 24: Maximum BitRate vs SNR for BER=0.01

From Figure 20-21, it is observed that maximum bit rate increases with SNR for a given BER.

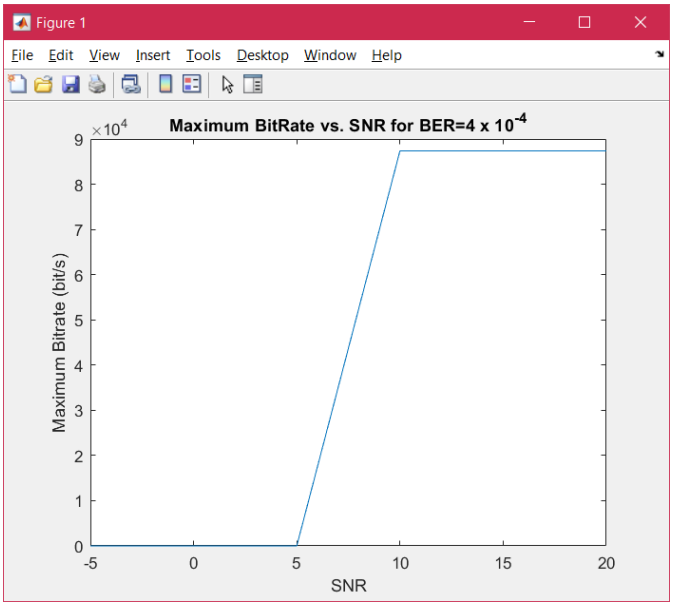


Figure 25: Maximum BitRate vs SNR for BER=0.0004

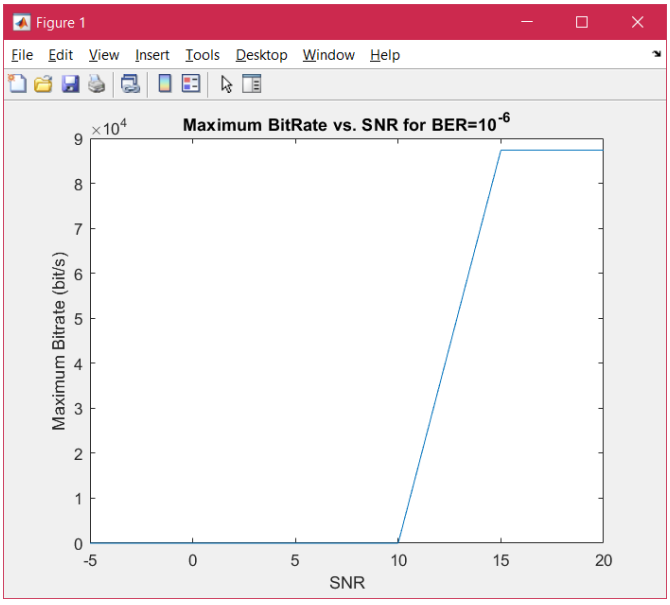


Figure 26: Maximum BitRate vs SNR for BER=0.000001

From Figure 22-23, it is observed that maximum bit rate increases with SNR for a given BER.

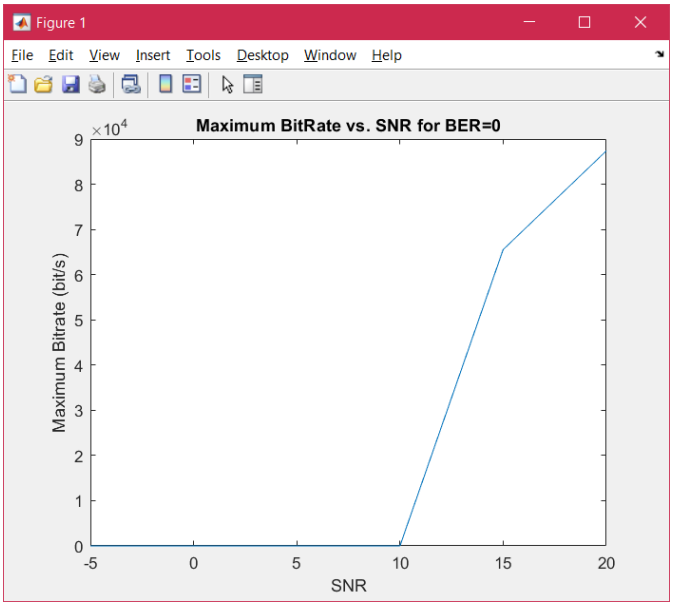


Figure 27: Maximum BitRate vs SNR for BER=0

From Figure 24 (at the left), it is observed that maximum bit rate increases with SNR for a given BER.

For the plots , I have filled the unavailable data with zero, as it is stated in the lab manual or completed it with the minimum value. For example, in Figure 18, I did not set the unavailable data points to 0. Otherwise, it would infer that maximum bitrate falls with increasing SNR. Instead, I have kept the value constant, which is not wrong since it sets a minimum value (which is the best I can do with non-available data points).

Since I work with the 256-QAM scheme, bitrate is already large for all cases. This is due to the number of bits each symbol carries. This is an advantage which makes 256-QAM (or larger QAMs) superior in bitrate. During the simulation, it may seem like 256-QAM is a bad scheme that we cannot increase bitrates in most cases. One may compare this situation with 4-QAM and think that it is easier to increase the bitrate of 4-QAM for a given SNR. However, instead of focusing on lower QAM modulations' (or other inferior) increasibility, one should focus on the final bitrates achieved. Then, it will be observed that by increasing lower order QAM's bitrate one only gets closer to the 256-QAM's hard-to-increase bitrate value. Consider an example of 4-QAM and 256-QAM comparison, for example. 4-QAM carries 2 bits/symbol whereas 256-QAM carries 8 bits/symbol. Therefore, by quadrupling its bitrate, 4-QAM scheme only gets equal to the 256-QAM case, in the expense of increasing its bandwidth, which is undesired.

TABLE: AWGN Setting Experiments

BER	SNR	M	T_s^{-1}	Bitrate (8/Ts)	Const. Type and Size	Min. Distance Between Symbols	Carrier Sample Frequency (Hz)	Raised Cosine Rolloff Factor	Raised Cosine Filter Span	Raised Cosine Output Samples /Symbol	Down-sample Factor
0.27	-5	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
0.26	0	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
	5	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
0.25	10	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
	15	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
	20	2	2^{14}	$\mathbf{2^{17} = 131072}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	2	2
10^{-1}	-5	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3
$7\ 10^{-2}$	0	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3
10^{-2}	5	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3
$4\ 10^{-4}$	10	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3
10^{-6}	15	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3
0	15	4	2^{13}	$\mathbf{2^{16} = 65536}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	4	4
	20	3	$\frac{2^{15}}{3}$	$\frac{\mathbf{2^{18}}}{\mathbf{3}} \cong \mathbf{87381}$	256 QAM - Size: 15	2	2^{-15}	0.2	10	3	3

Table 5: AWGN Setting Experiment

Overall, in this part we have observed that as SNR increased - even though highest bitrate for that SNR did not change – BER values for that bitrate changed. Therefore, for the same bitrate, it is confirmed that lower SNR value causes more BER whereas higher SNR's are much safer. This can be seen the other way too: For a given channel (fixed SNR), BER increases as we increase the bitrate (with the same constellation scheme). The third way of seeing this result is that if we want to limit our application with a fixed BER, more SNR means less bitrate. Either SNR must be lowered (channel modification), or bitrate must be increased (either by increasing bits/symbol by changing the scheme or decreasing Ts).

BONUS PART: DISPLAY CORRECTION

Consider the 256-QAM system, whose Simulink model is as follows:

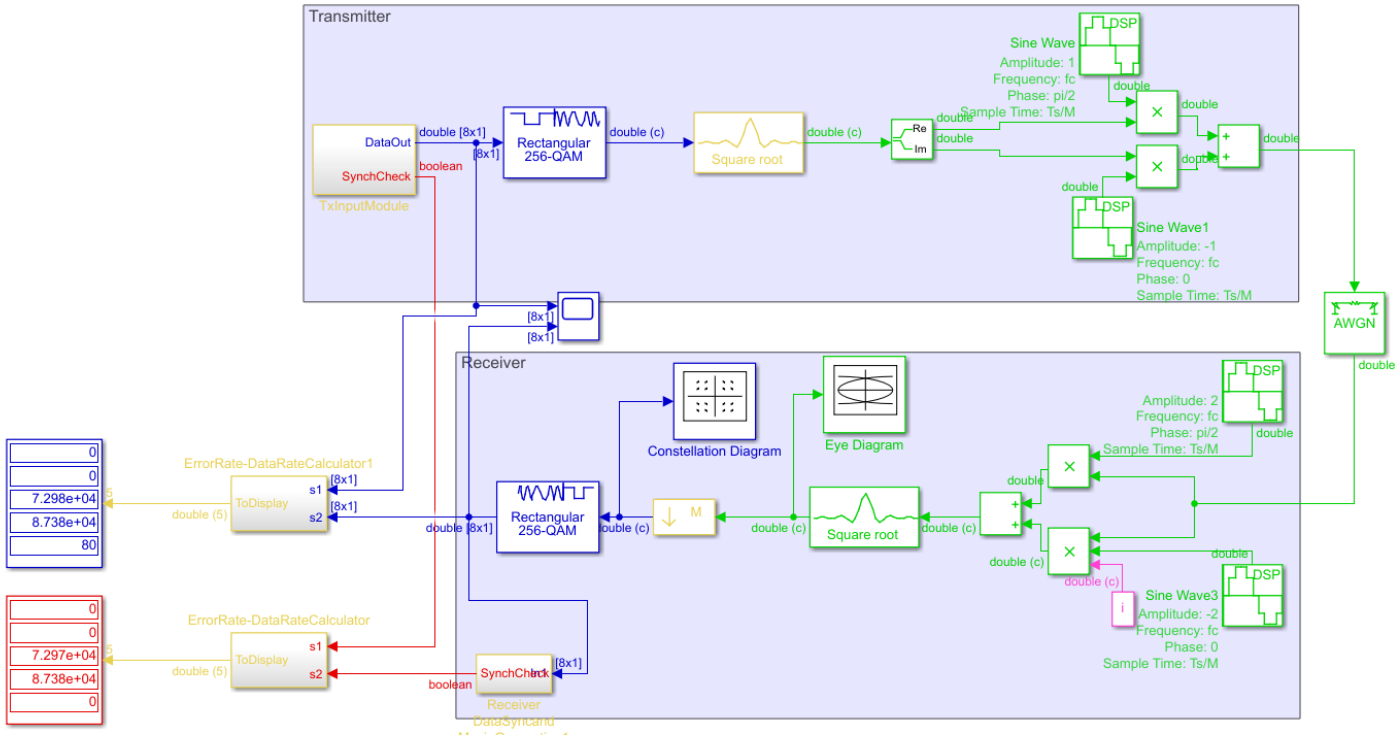


Figure 28: Simulink model for the first bonus

The properties and most of the necessary screenshots are already provided in the preceding section. Here, there is an AWGN block added, which is not important in this case because it does not cause any delay. The colorings will be helpful for explanation, so please pay attention to the colors.

The problem with the display can be demonstrated as follows:

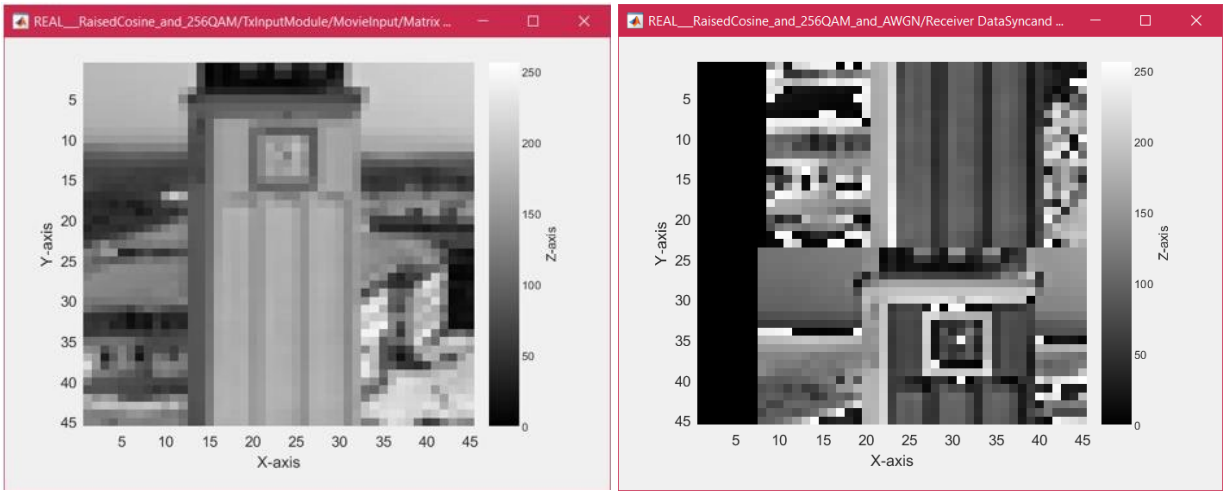


Figure 29: Transmitted image view (left) vs. Received image view (right)

The matrix view of the received image has both color pixel and hue problems, along with image's being *drifted.* The image drift problem can be easily explained by delay. Matrix view block is fed by pixels, where the pixels form a 45x45 image at the end, with each pixel is being constructed from 8-bit unsigned numbers. This is what happens in the GenerateMovie block. This is not something I have implemented, nor modified. What happens inside is as follows by order:

i. Buffer takes bits and converts each group of 8 elements into an 8x1 matrix.
ii. This 8-element matrix is fed to Bit to Integer Converter, where corresponding 8-bit unsigned integer is constructed.
iii. inpmov is a 45x45 movie. When $45 \times 45 = 2025$ pixels (numbers in the range $\{0,2^8 - 1\}$) are formed at the output of Bit to Integer Converter block, buffer takes them and form a 2025x1 matrix. This is a frame of the movie.
iv. This frame is reshaped by the Reshape block to a 45x45 matrix, and then transferred to Matrix Viewer so we see it on the pop-up screen.

So, what we need to do is to give enough time to the GenerateMovie block to construct a frame. Delay block at the input of the GenerateMovie block does that. It delays the signal, until the block is enabled after the SynchCheck and enough number of bits are received to form a frame of the movie. However, there are few issues here. Consider the case where after GenerateMovie is enabled, Buffer2 inside GenerateMovie block has just accumulated 45x45x8 *movie signals*. Then, first frame of the movie would be shown just after SynchCheck is completed. Now, suppose timing is not that perfect, and after SynchCheck is completed, 45x8 empty signals are arrived, and then movie signals are arrived. Then, since 45x45x8 bits are accumulated (45x8 of them are non-movie 0s), this would be shown on matrix. What we would see is a black line, and the first frame at the rest of the window. After another 45x45x8 bits arrive, then the 45x8 bits from the previous frame would be shown on the next frame. Well, this is problematic, and this is what we try to avoid with the Delay block at the input of GenerateMovie block. We observe the case with no a delay of 0, and then insert the necessary amount of delay, which is 22x8=176 in this case. But this does not solve the hue and pixel problem and leaves us with this:

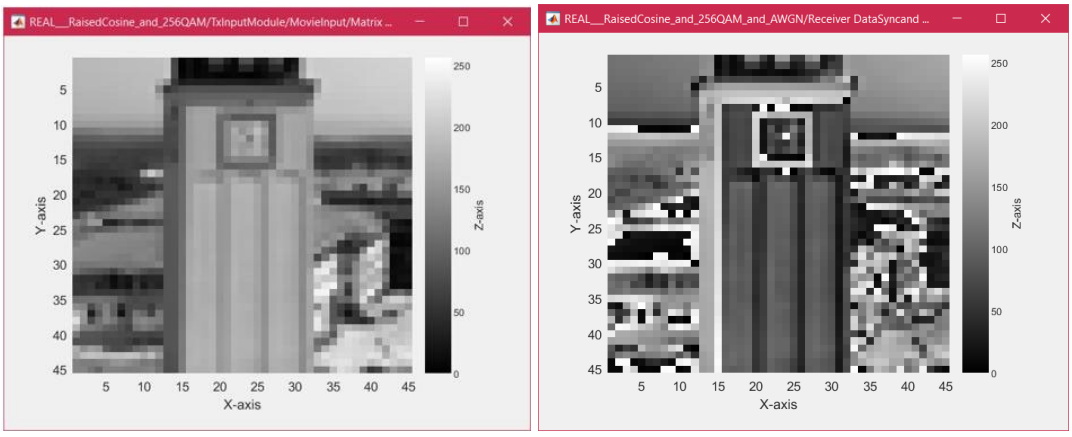


Figure 30: Transmitted image view (left) vs. Received image(justified) view (right)

To solve this hue and pixel problem, let us focus on the signal properties. First, examine the MovieInput module inside TxInputModule, where the inpmov comes into play in the Simulink model.

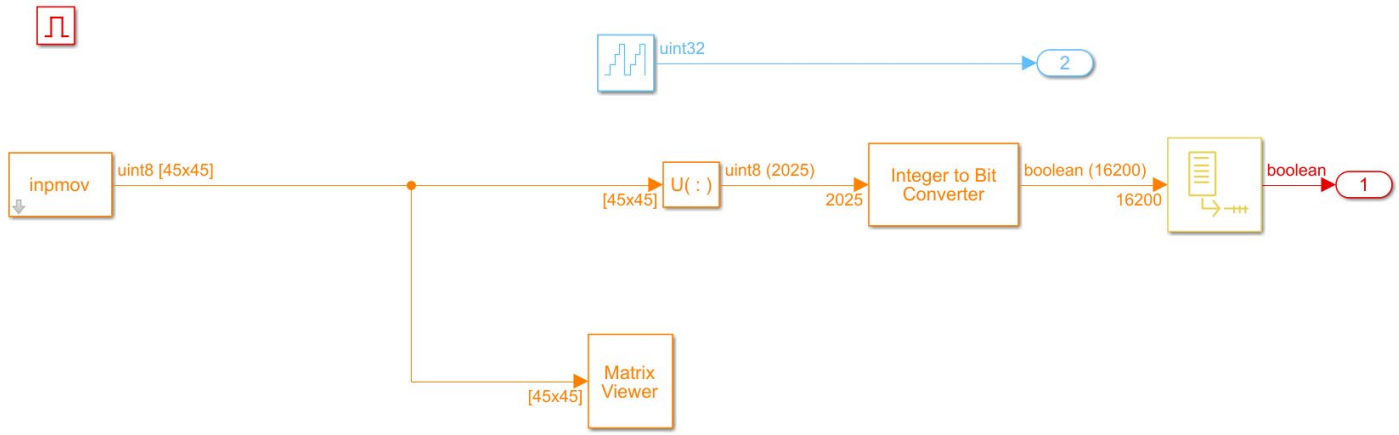


Figure 31: MovieInput Block

Let us trace the sampling of the movie signal along the way. The movie signal inpmov - which is a 45x45x100 uint8 matrix provided by the instructor - is imported at the very most left block (Signal From Workspace block). Then, at one arm, it is directly uploaded to the matrix viewer. Note that this block has a Sample Time. Therefore, it loads each frame at the integer multiples of these sample time. The other arm goes into Reshape block, which eventually goes to the output of the module. Following this path, operations performed on it are **reshaping, integer to bit conversion** and **unbuffering.**

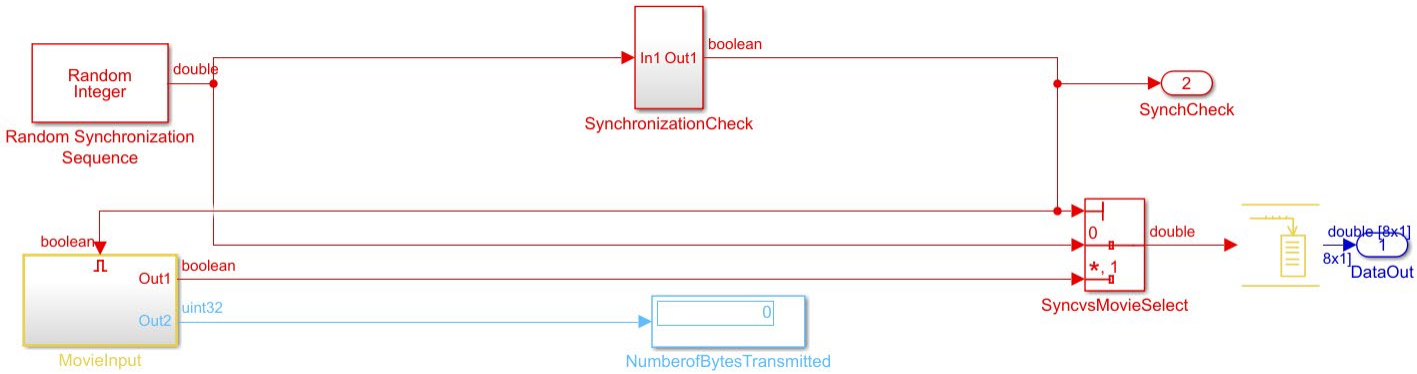


Figure 32: TxInputModule

After the movie signal leaves MovieInput block, it is subjected to a downsampling (buffering) before it leaves the TxInput Module. Here it is **buffered** once. This operation creates a 8 parallel signals (8x1 matrix) since output buffer size is set to log2(ConstellationSize). Then, the movie signal leaves this module. Follow the rest of its journey from Figure 16. Raised cosine block upsamples the signal, but then it is downsampled back via Downsample block in the receiver. So, its effect is eliminated. Then, it enters Receiver DataSyncand MovieGeneration1 block.

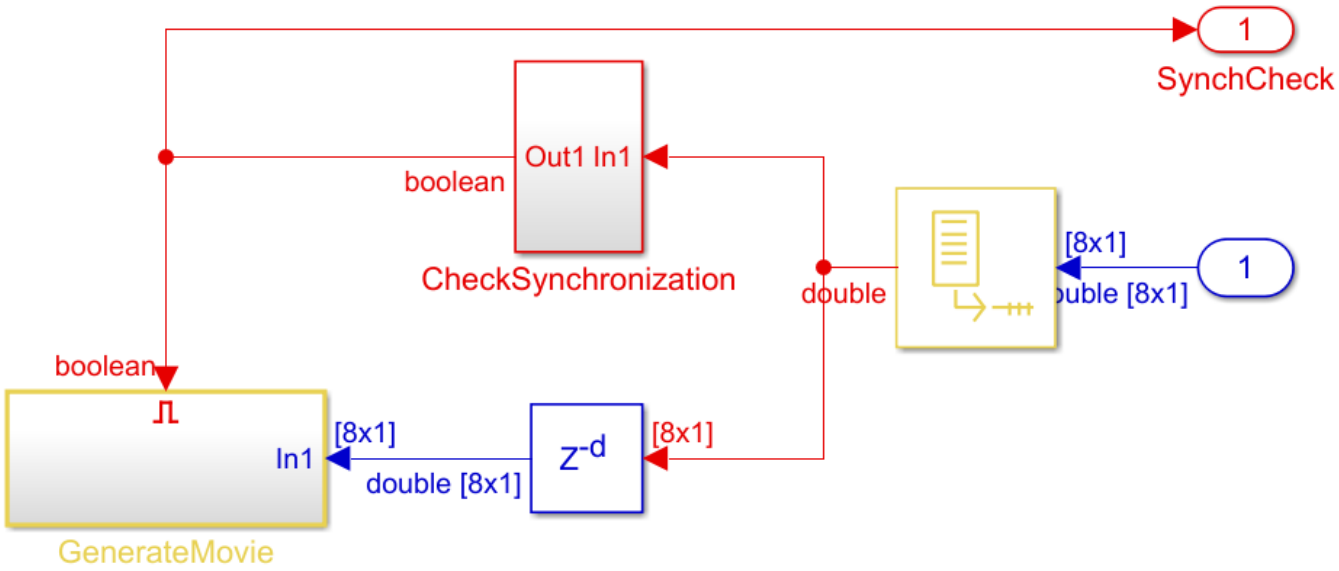


Figure 33: Receiver DataSyncand MovieGeneration1

Here, there is first **unbuffering.** This unbuffering compensates the ***buffering*** at the end of the TxInputModule. This is fine for the SychCheck output because the sampling rate of it is now matches the sampling rate of the SynchCheck output of the TxInputModule (See Figure 20 and pay attention to the colors). Suppose we have dealt with the delay problem and move onto the GenerateMovie block.

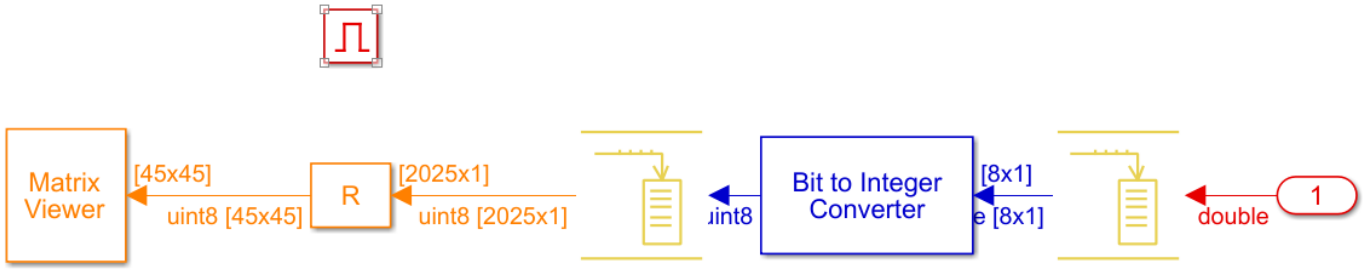


Figure 34:GenerateMovie Block

In this block, there are **2 buffering, 1 bit to integer conversion** and **1 shaping** operation performed on the movie signal. Now, to understand this part better, we go back. Input of Figure 30 is parallel bit sequences of 8-bit sequences. In other words, suppose $D8_1$ is an 8-bit sequence. Then, input is $[D8_1;\ D8_2; \dots]$. Unbuffering it puts it into form $[D8_1, D8_2, \dots]$. Then, when it goes through the first buffer in Figure 31, it becomes $[D8_1;\ D8_2; \dots]$. This is a matrix, not a vector. However, Bit to Integer block works with vectors and outputs a scalar value. So, I believe that it performs the operations to the whole matrix, which in returns generates a wrong color value (pixel). To prevent this from happening, output of GenerateMovie Block must be the parallel bit sequences of 8-bit sequences, in other words, $[D8_1;\ D8_2; \dots]$. Then, buffer in Figure 31 performs on each row, which in result generates a column vector (1D). Then, this is transferred into Bit to Integer Converter. Since it is now a vector (not a matrix), Bit to Integer Converter works as it should be, and generates the correct pixels. Note that this is *not* a fatal error, we somehow get the movie signal up to a great rate. However, it causes a hue problem. The solution to it is to remove this unbuffering for the lower arm in Figure 21. The solution I suggest, and that works is as follows:

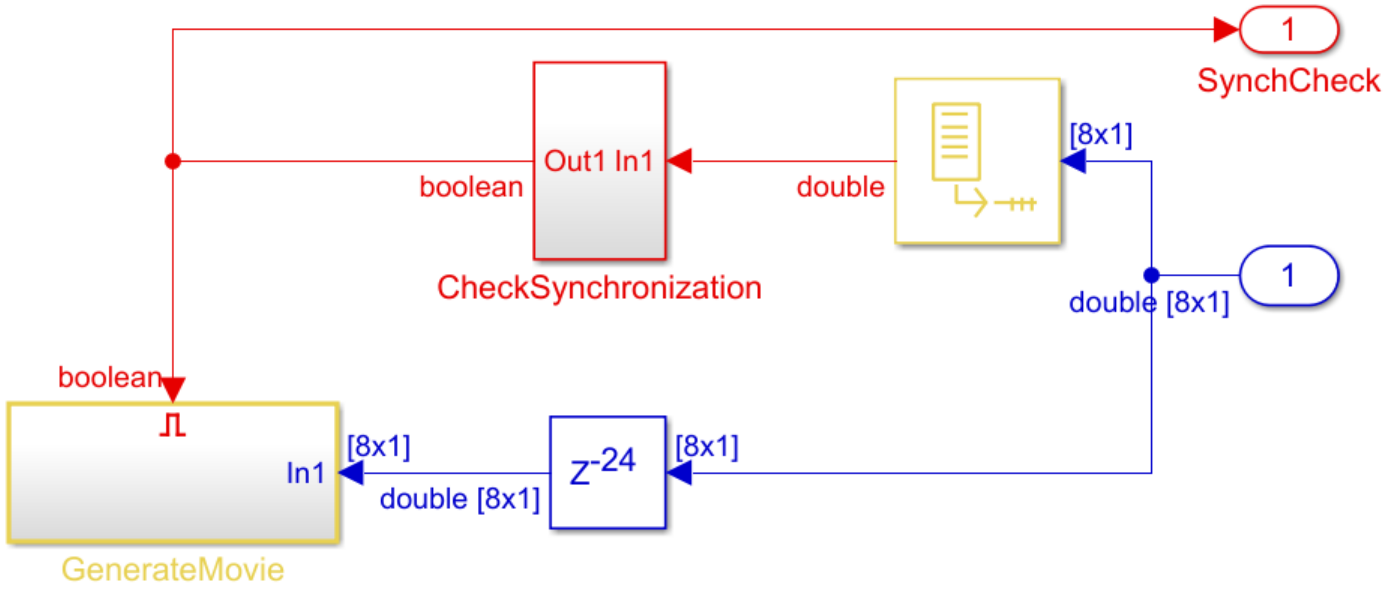


Figure 35: Modified Receiver DataSyncand MovieGeneration1

With this modification, problems I have explained are solved. Note that I have also changed the delay of the Delay block after this operation. This is due to the fact that now delay caused by the unbuffer block does not occur. As a result, the delay required drops down to 24. See the outputs I receive for this setup:

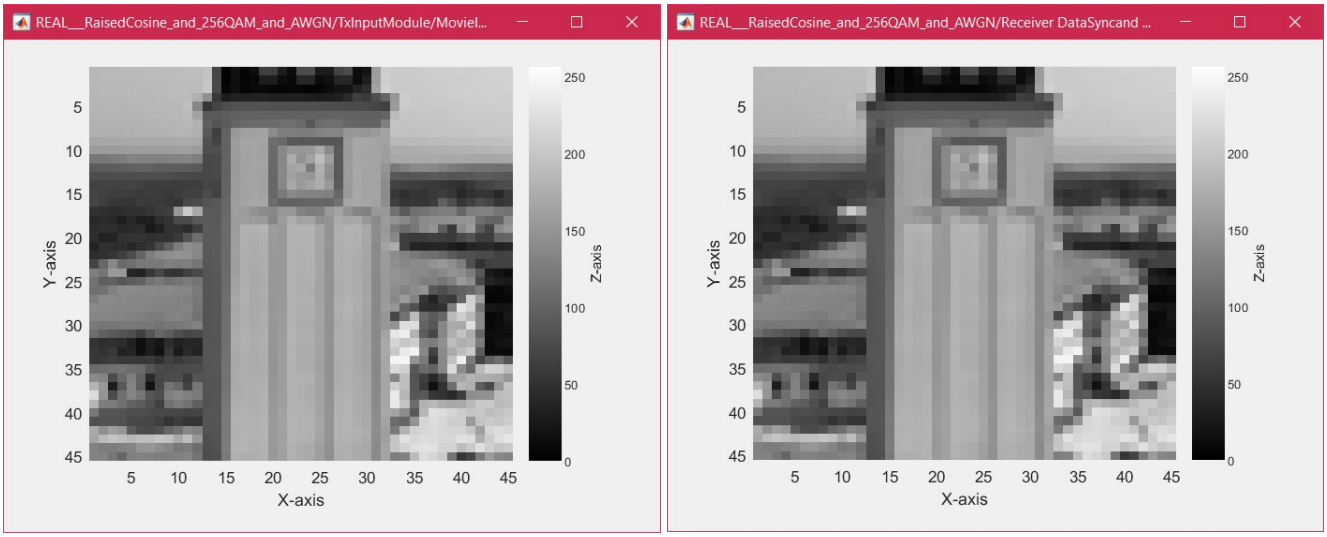


Figure 36: Transmitted image view (left) vs. Received image(justified and modified) view (right)

Observe that now both the transmitted and received image is identical (in the case of BER of 0 of course). I have mostly mentioned about my observations throughout the explanations. In short, the most important points for me were:

- Delay caused by unbuffer block.
- Notion of sampling – upsampling and how it should be compensated.
- Serial – parallel conversions and blocks acting on rows or columns.
- How delay may cause the corruption of the image.
- How upsampling – downsampling may cause the distortion of the image.

CONCLUSIONS

In this project, I have first transmitted the given movie signal at different upsampling rates. In this part, the importance of M (how many times the data points are to be repeated / filled with 0s between) is clearly demonstrated. To summarize, I have demonstrated 3 cases there:

1) No upsampling → δ pulses in time domain caused high interference in time due to their being infinite in time. In general, δ pulses are good for multiplication in the *frequency domain*, not in the time domain. Otherwise, one must deal with convolutions in the frequency domain with an infinite BW function.
2) Undersampling → If we do not upsample with enough taps, upsampling fails. The more we upsample, the better we give pulse a shape. The less we upsample, the more the signal looks and behaves like a δ. Not enough is unwanted, but too much is also not efficient. Therefore, a balance is sought.
3) Upsampling at the right rate → This was the last case, but also the only case we recovered signal back with a good precision. Actually, the signal is completely got back with a BER of 0.

M is also related to the bitrate, in fact, its relation to bitrate is what we are interested in the most. In this project, we are asked to keep T_s/M ratio constant. This means is we are to increase or decrease M, we also effect T_s. This is why using unnecessarily large M is not preferred: when M is decreased, T_s is decreased. When T_s is decreased, $1/T_s$ is increased so that the transmission rate gets faster. For a fixed constellation scheme (256-QAM was my fixed constellation in this project), increasing bitrate is only possibly by increasing $1/T_s$. So, finding the balance between M and quality of the signal is important if we are to increase the efficiency of the transmission by increasing the bitrate with as less error as possible.

The AWGN Bonus part was similar to the main task of the project. This time we have put a AWGN block between the transmitter and receiver, which is something we have done many times this semester in this lecture, both physically and virtually. Depending on the SNR ratio, bitrate is tried to be maximized. I could not observe significant changes for fixed BER values; however, it was clear that bitrate drops with decreased SNR significantly. I have explained some of the perspectives we can gain from this experiment in the end of the section. Overall, it was a challenging experiment.

The Display Correction Bonus was my favorite part in this project. It required a great effort to understand the blocks provided to us. I have tried to interpret them and understand their functionalities as much as possible. This part was consisted of 2 objectives. One of them was to deal with the color problem, and the other one was to deal with the image shift issue. For me, both of them were very challenging. In the end, it took me less time to solve the image shift problem when compared to the hue (color) problem. I think this is the case for many. It is more natural to think that image is shifted due to delay, rather than thinking upsample – downsample incompatibility issue caused image pixels to be inaccurately constructed. In this part, the importance of delay is once again very well understood. Furthermore, I have expanded my knowledge on Simulink blocks. As an extra, to be able to work out the problem faster, I needed to increase the bitrate so that system would produce outputs faster. So, this part was also a bit rate increase challenge.

Unfortunately, I have not been able to implement the equalizer. In fact, I could not spend as much time as I want on it due to other projects and homework. However, the concept of equalizer is understood, even though the equalizer block is provided to us is a very much complex one and requires more time to think on it to understand its functionality. I believe equalizer could be helpful to get the target BER value which are missed in between. Overall, this was an educative and challenging project, especially the last part. I believe that bitrate – constellation size – upsampling – Ts relations are clear in my mind now. I hope my report is also clear and comprehensible enough too.