DIGITAL COMMUNICATION

Report on Course Project

W.A.A.D.Jayasekara

ENG-17-068

Batch of 2021

Faculty of Engineering

Sri Lanka Technological Campus

Acknowledgement

I have received an abundance of such assistance, for which I am deeply grateful. For their generous help, I want to thank **Mr. Selvakumar Tharranetharan** whose valuable guidance and kind supervision given to me throughout the **Digital Communication** module.

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

Abstraction

Digital communication is the study of how a analog message is transformed into a digital and send via medium to another location and receiver the analog data.

- Digital communication is any message passed through digital devices.
- Digital communication is any type of information sent digitally.

Table of Contents

Table of Figures

Objectives

The objectives of this project activity are to give students experience with digital communications processes such as source coding, channel coding, modulations, channel impacts, demodulation, channel decoding and source decoding via a MATLAB simulation and compare the performance of such various techniques.

Introduction

Digital communication is a mode of communication where the information or the through is encoded digitality as discrete signals and electronically transtered to the recipients,we were learnt about source coding ,channel coding and modulation,demodulation etc. information theory and mathematical behaviors of these techniques in digital communication.

We were using MATLAB software to perform our task and plotted the graphs according to the tasks. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Task

A rgb image is given to you. You have to convert the rgb image into bits and transmit via an additive white Gaussian noise channel. Perform the following tasks.

Task 1: Transmit the rgb image with 2-ASK, 4-ASK, 8-ASK modulation technique over AWGN channel with $\frac{\text{Eb}}{\text{N0}}$ = 5 dB and compare the original image and the received images.

Step-1: Load an rgb image in matlab, get it parameters and display it.
Step-2: Convert the image into bits and display first 100 bits.
Step-3: Transmission via noise channel.

Then plot the graphs using MATLAB graph function.

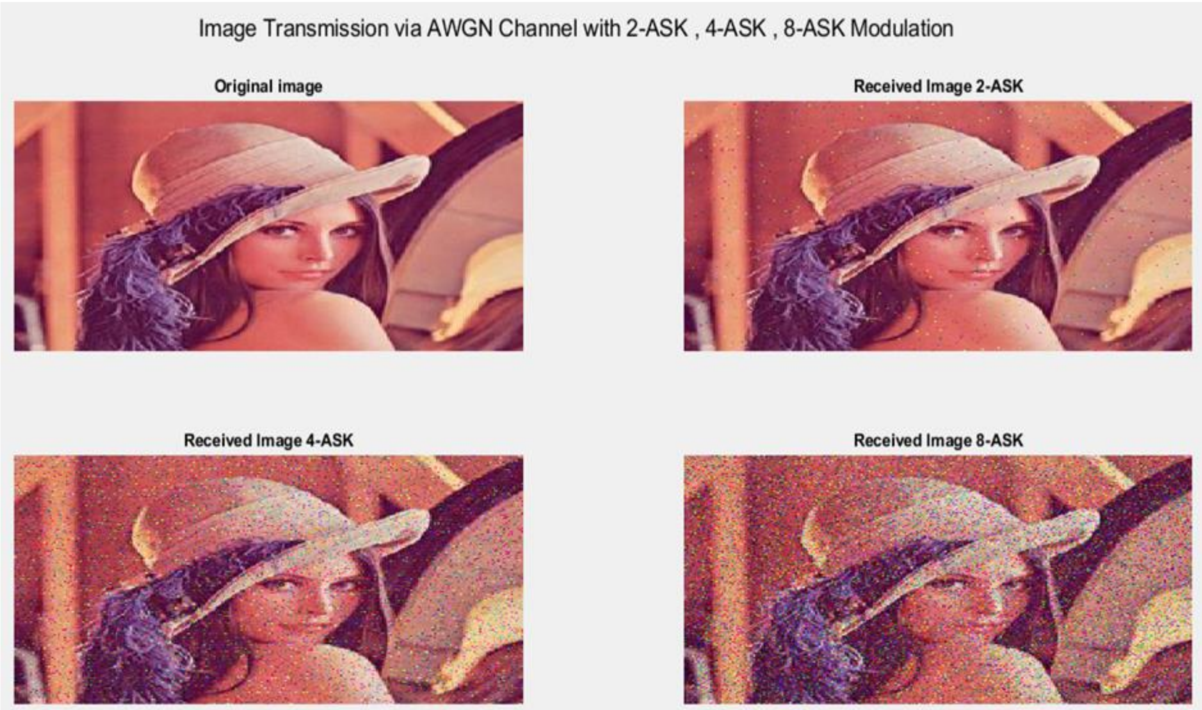

Figure 1-Image Transmission via AWGN Channel with 2-ASK , 4-ASK , 8-ASK Modulation

Task 2: Transmit the rgb image with 2-ASK with different channel codes (convolution code raten $\frac{1}{2}$, convolution code rate $\frac{2}{3}$, repetition code 3 and repetition code 5) over AWGN channel with $\frac{\text{Eb}}{\text{N0}}$ = 5 dB and compare the received images. (You need to modify the channel coder and channel decoder function to use repetition code 5 codes. Other three codes are already done for you.)

Step 1: first setp is to convert to simple ratio.
Step 2:transmit the bits through the AWGN channel using modulation with channel coding.(calling this function we have to select the coding type and similar energy levels.)

Figure 2-Impact of Channel Coding in Transmission

Task 3: A function plot_ber2ask is given to compare the simulated BER of 2-ASK and theoretical BER of 2-ASK. Create similar function to plot the simulated BER of 4-ASK and 8-ASK. Then, compare the simulated BERs of 2-ASK, 4-ASK, and 8-ASK modulations in the range of $\frac{\mathrm{Eb}}{\mathrm{N0}}$ = -4 dB to $\frac{\mathrm{Eb}}{\mathrm{N0}}$ = 10 dB in a single plot. (No need to plot the theoretical BERs)

Step-1: Compare the BER of 2-ASK, Theoretical BER of 2-ASK and compare BERs of 2-ASK.
Step 2: Transmit the modulated signals and received the signal through the AWGN channel.
Step-3: Demodulate and convert those symbols to bits.
Step 4: Make the BER graphs using these received bits and transmitted bits.
Step-5: Finally the graphs of BERs between Eb/No = -4 to Eb/No = 10.

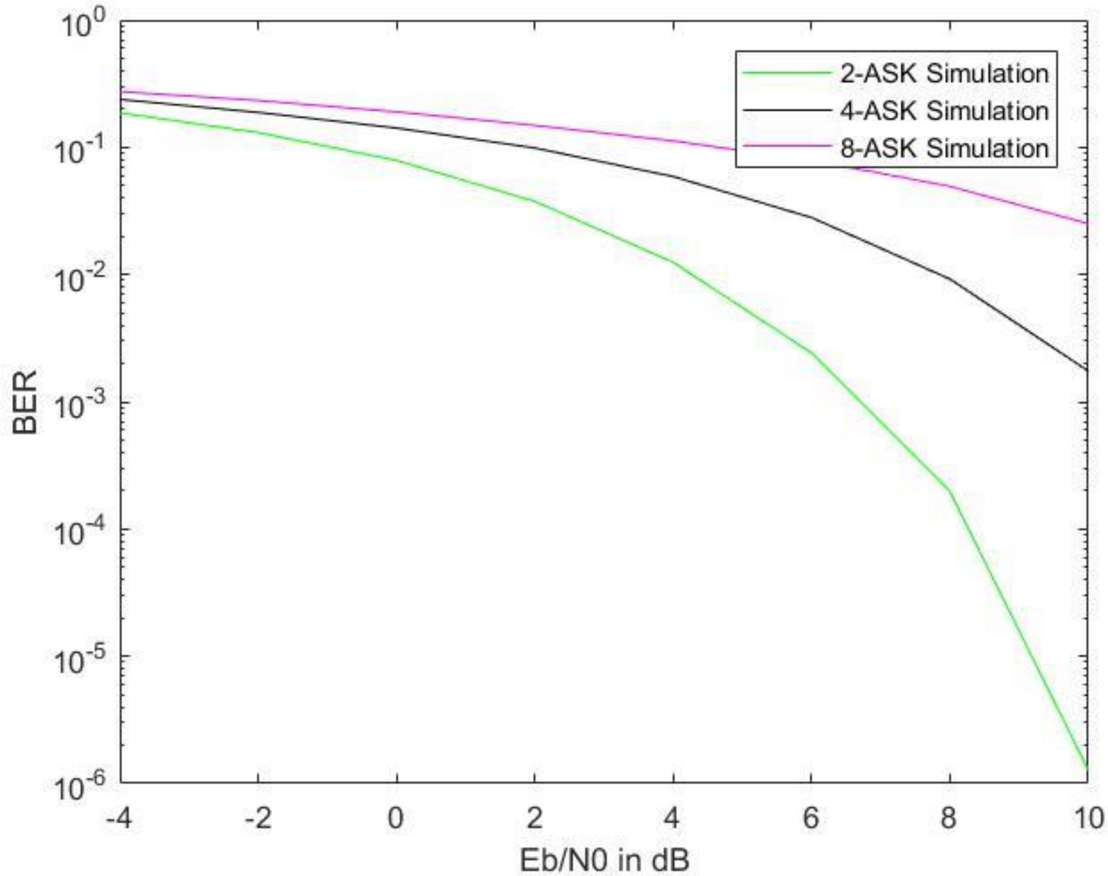

Figure 3-Plot of the simulation BERs of 2-ASK, 4-ASK, and 8-ASK modulations

Task 4: Write a similar function in Task 3 for to plot the BER vs $\frac{\text{Eb}}{\text{N0}}$ in dB for channel coded transmission with rate = $\frac{1}{2}$ convolution code with 2-ASK modulation. Then, compare the BER plots of rate $=\frac{1}{2}$ convolution 2-ASK and 2-ASK without channel coding.

Step 1: Reconstruct the function above use as in this function.
Step 2: Created the graph comparing the 2-ASK modulation with channel coding and without channel coding.

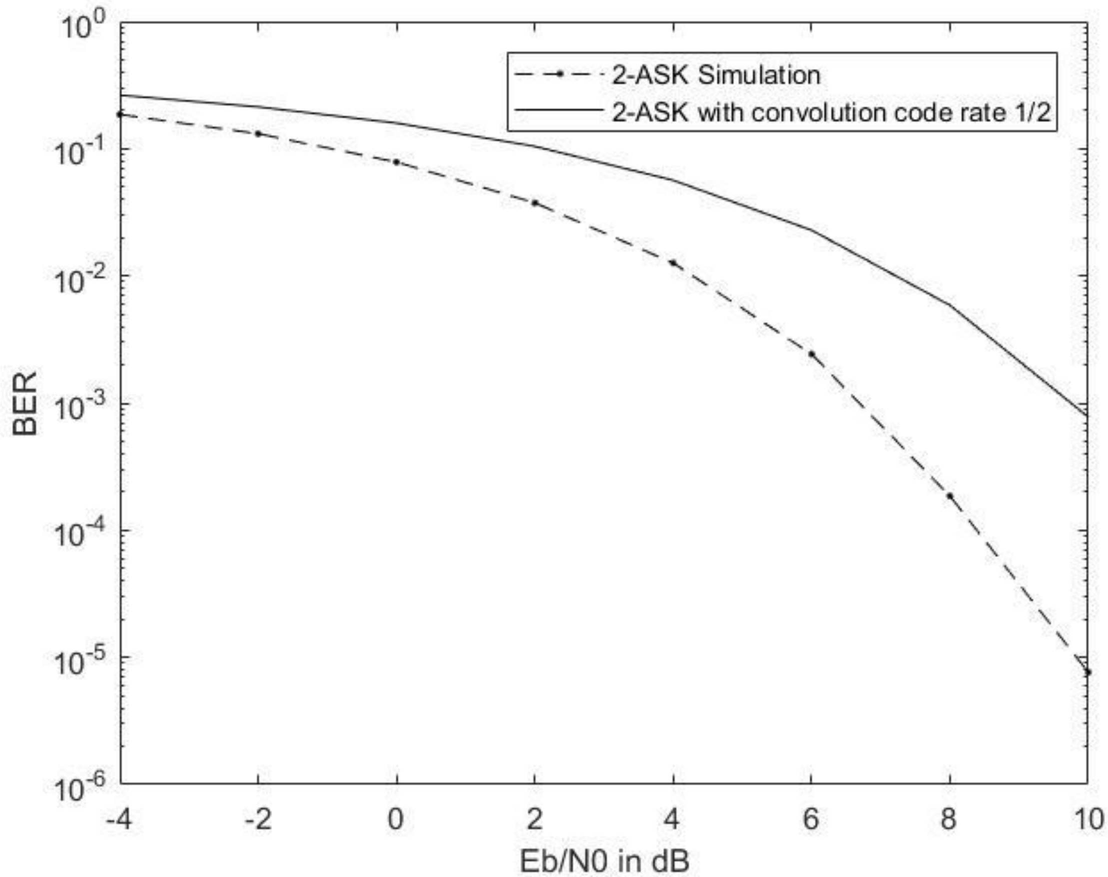

Figure 4- BER plots of convolution 2-ASK and 2-ASK without channel coding

Task 5: Briefly explain the reasons behind the above plots (Why the bit error rate curves are very like that).

In digital communication, the information transmit the using 1 and 0 bits. During the transmission due to the other interference and other reasons, can be added the noise to the original message and it will effect to receive the original message.

BER is the one way of get the information of about errors that effected to the original message. The bit error rate (BER) is the number of bit errors divided by the total number of transmitted bits over a channel. BER although unit-less also expressed in terms of percentage

. Eb/No is classically defined as the ratio of Energy per Bit (Eb) to the Noise Power (No). Eb/No is the measure of signal to noise ratio for a digital communication. It is measured at the input to the receiver.

Appendix

Task 1 Source code

```
clc; %clear the command window
clear all;  %clear workspace

%%

ImageA=imread('lena256.jpg');
[len,wid,hei]=size(ImageA);
bits_per_value=8;
fprintf(' Length : %d\n Width  : %d\n Height : %d\n', len,
wid, hei);
fprintf('(The length and the width are defined by the image
pixels and height is 3 for RGB images)\n');

figure
image(ImageA);
title('Imported original image');
axis off;

fprintf('Imported  image  \n .. \n .. \n .. ');

fprintf('Displaying Imported Image. \nProgram paused. Press
enter to Next Step..\n');
pause;

%%

fprintf('\n Wait! It might be take some time \n Image to
bits conversion in progress... \n');
bits=image2bits(ImageA, len, wid, hei, bits_per_value);
fprintf('\n First 100 bits: ');
fprintf('%d', bits(1:100));
fprintf('\n');
fprintf('\nProgram paused. Press enter to Next Step..\n');
pause;

%%
```

```
fprintf('\n Wait! It might take some time \n bits to Image
conversion in progress... \n');

Eb_N0_dB=5;
Eb_N0=10^(Eb_N0_dB/10);

k=1.38*10^-23;
T=300;
N0=k*T;

Eb=Eb_N0*N0;

Noise_power=N0/2;

Es_2ASK=Eb;
Es_4ASK=2*Eb;
Es_8ASK=3*Eb;

Transmitted_symbols_2ASK=sqrt(Es_2ASK).*askmodulation(bits,
2);
Transmitted_symbols_4ASK=sqrt(Es_4ASK/5).*askmodulation(bit
s, 4);
Transmitted_symbols_8ASK=sqrt(Es_8ASK/21).*askmodulation(bi
ts, 8);

Gaussian_noise_2ASK=0+sqrt(Noise_power).*randn(length(Trans
mitted_symbols_2ASK), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbols_2AS
K), 1);

Gaussian_noise_4ASK=0+sqrt(Noise_power).*randn(length(Trans
mitted_symbols_4ASK), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbols_4AS
K), 1);

Gaussian_noise_8ASK=0+sqrt(Noise_power).*randn(length(Trans
mitted_symbols_8ASK), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbols_8AS
K), 1);
```

```
Received_symbols_2ASK=Transmitted_symbols_2ASK+Gaussian_noi
se_2ASK;
Received_symbols_4ASK=Transmitted_symbols_4ASK+Gaussian_noi
se_4ASK;
Received_symbols_8ASK=Transmitted_symbols_8ASK+Gaussian_noi
se_8ASK;

Normalized_symbols_2ASK=Received_symbols_2ASK./sqrt(Es_2ASK
);
Normalized_symbols_4ASK=Received_symbols_4ASK./sqrt(Es_4ASK
/5);
Normalized_symbols_8ASK=Received_symbols_8ASK./sqrt(Es_8ASK
/21);

Received_bits_2ASK=askdemodulation(Normalized_symbols_2ASK,
2);
Received_bits_4ASK=askdemodulation(Normalized_symbols_4ASK,
4);
Received_bits_8ASK=askdemodulation(Normalized_symbols_8ASK,
8);

ImageC=bits2image(Received_bits_2ASK, len, wid, hei,
bits_per_value);
ImageD=bits2image(Received_bits_4ASK, len, wid, hei,
bits_per_value);
ImageE=bits2image(Received_bits_8ASK, len, wid, hei,
bits_per_value);

figure
subplot(2,2,1); image(ImageA); title('Original image');
axis off;
subplot(2,2,2); image(ImageC); title('Received Image 2-
ASK'); axis off;
subplot(2,2,3); image(ImageD); title('Received Image 4-
ASK'); axis off;
subplot(2,2,4); image(ImageE); title('Received Image 8-
ASK'); axis off;
suptitle('Image Transmission via AWGN Channel with 2-ASK ,
4-ASK , 8-ASK Modulation');
fprintf('\nProgram paused. Press enter to END');
pause;
```

Task 2 Source code

```
clc; %clear the command window
clear all;  %clear workspace

%%

ImageA=imread('lena256.jpg');
[len,wid,hei]=size(ImageA);
bits_per_value=8;
fprintf(' Length : %d\n Width  : %d\n Height : %d\n', len,
wid, hei);
fprintf('(The length and the width are defined by the image
pixels and height is 3 for RGB images)\n');

figure
image(ImageA);
title('Imported original image');
axis off;

fprintf('Imported  image  \n .. \n .. \n .. ');

fprintf('Displaying Imported Image. \nProgram paused. Press
enter to Next Step..\n');
pause;

%%

fprintf('\n Wait! It might take some time \n Image to bits
conversion in progress... \n');
bits=image2bits(ImageA, len, wid, hei, bits_per_value);
fprintf('\n First 100 bits: ');
fprintf('%d', bits(1:100));
fprintf('\n');
fprintf('\nProgram paused. Press enter to Next Step..\n');
pause;

%%

fprintf('\n Wait! It might take some time \n Bits to Image
conversion in progress... \n');

Eb_N0_dB=5;
```

```
Eb_N0=10^(Eb_N0_dB/10); %Decibels to ratio conversion

k=1.38*10^-23;
T=300;
N0=k*T;

Eb=Eb_N0*N0;

Es_coded_2ASK=Eb;
Ecb_1_2=0.5*Eb;
Ecb_2_3=(2/3)*Eb;
Es_coded_2ASK_1_2=Ecb_1_2;
Es_coded_2ASK_2_3=Ecb_2_3;
Noise_power=N0/2;

coded_bits_1_2=channel_coder(bits,'conv_R=1_2');
coded_bits_2_3=channel_coder(bits,'conv_R=2_3');
coded_bits_Rep3=channel_coder(bits,'repetition3');
coded_bits_Rep5=channel_coder(bits,'repetition5');

Transmitted_symbolsC_1_2=sqrt(Es_coded_2ASK_1_2).*askmodula
tion(coded_bits_1_2, 2);
Transmitted_symbolsC_2_3=sqrt(Es_coded_2ASK_2_3).*askmodula
tion(coded_bits_2_3, 2);
Transmitted_symbolsC_Rep3=sqrt(Es_coded_2ASK).*askmodulatio
n(coded_bits_Rep3, 2);
Transmitted_symbolsC_Rep5=sqrt(Es_coded_2ASK).*askmodulatio
n(coded_bits_Rep5, 2);

Gaussian_noiseC_1_2=0+sqrt(Noise_power).*randn(length(Trans
mitted_symbolsC_1_2), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsC_1_
2), 1);
Gaussian_noiseC_2_3=0+sqrt(Noise_power).*randn(length(Trans
mitted_symbolsC_2_3), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsC_2_
3), 1);
Gaussian_noiseC_Rep3=0+sqrt(Noise_power).*randn(length(Tran
smitted_symbolsC_Rep3), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsC_Re
p3), 1);
```

```
Gaussian_noiseC_Rep5=0+sqrt(Noise_power).*randn(length(Transmitted_symbolsC_Rep5), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsC_Rep5), 1);

Received_symbolsC_1_2=Transmitted_symbolsC_1_2+Gaussian_noiseC_1_2;
Received_symbolsC_2_3=Transmitted_symbolsC_2_3+Gaussian_noiseC_2_3;
Received_symbolsC_Rep3=Transmitted_symbolsC_Rep3+Gaussian_noiseC_Rep3;
Received_symbolsC_Rep5=Transmitted_symbolsC_Rep5+Gaussian_noiseC_Rep5;

Normalized_symbolsC_1_2=Received_symbolsC_1_2./sqrt(Es_coded_2ASK_1_2);
Normalized_symbolsC_2_3=Received_symbolsC_2_3./sqrt(Es_coded_2ASK_2_3);
Normalized_symbolsC_Rep3=Received_symbolsC_Rep3./sqrt(Es_coded_2ASK);
Normalized_symbolsC_Rep5=Received_symbolsC_Rep5./sqrt(Es_coded_2ASK);

Received_coded_bitsC_1_2=askdemodulation(Normalized_symbolsC_1_2, 2);
Received_coded_bitsC_2_3=askdemodulation(Normalized_symbolsC_2_3, 2);
Received_coded_bitsC_Rep3=askdemodulation(Normalized_symbolsC_Rep3, 2);
Received_coded_bitsC_Rep5=askdemodulation(Normalized_symbolsC_Rep5, 2);

Received_bitsC_1_2=channel_decoder(Received_coded_bitsC_1_2,'conv_R=1_2');
Received_bitsC_2_3=channel_decoder(Received_coded_bitsC_2_3,'conv_R=2_3');
Received_bitsC_Rep3=channel_decoder(Received_coded_bitsC_Rep3,'repetition3');
Received_bitsC_Rep5=channel_decoder(Received_coded_bitsC_Rep5,'repetition5');

ImageE_1_2=bits2image(Received_bitsC_1_2, len, wid, hei, bits_per_value);
```

```
ImageE_2_3=bits2image(Received_bitsC_2_3, len, wid, hei,
bits_per_value);
ImageE_Rep3=bits2image(Received_bitsC_Rep3, len, wid, hei,
bits_per_value);
ImageE_Rep5=bits2image(Received_bitsC_Rep5, len, wid, hei,
bits_per_value);


figure
subplot(2,2,1); image(ImageE_1_2); title('AWGN Channel 2-
ASK with Coding Convolution code rate 1/2'); axis off;
subplot(2,2,2); image(ImageE_2_3); title('AWGN Channel 2-
ASK With Coding Convolution code rate 2/3'); axis off;
subplot(2,2,3); image(ImageE_Rep3); title('AWGN Channel 2-
ASK With Coding Repitition 3'); axis off;
subplot(2,2,4); image(ImageE_Rep5); title('AWGN Channel 2-
ASK With Coding Repitition 5'); axis off;
suptitle('Impact of Channel Coding in Transmission');
fprintf('\nProgram paused. Press enter to continue.\n');
pause;
```

Task 3 Source code

```
fprintf(' The length : %d\n The width  : %d\n The height :
%d\n', len, wid, hei);
fprintf('(The length and the width are defined by the image
pixels and height is 3 for RGB images)\n');

figure
image(ImageA);
title('Imported original image');
axis off;

fprintf('Imported  image  \n .. \n .. \n .. ');

fprintf('Displaying Imported Image. \nProgram paused. Press
enter to Next Step..\n');
pause;

%%

fprintf('\n Wait! It might take some time \n Image to bits
conversion in progress... \n');
bits=image2bits(ImageA, len, wid, hei, bits_per_value);
fprintf('\n First 100 bits: ');
fprintf('%d', bits(1:100));
fprintf('\n');
fprintf('\nProgram paused. Press enter to Next Step..\n');
pause;

%%

figure
plot_bertoask_t3(-4, 10, bits);
legend('2-ASK Simulation','4-ASK Simulation','8-ASK
Simulation');
xlabel('Eb/N0 in dB');
ylabel('BER');
```

“plot_bertoask_t3.m” function Source code

```
function plot_bertoask_t3(Eb_N0_dB_Start, Eb_N0_dB_Stop, 
bits)

k=1.38*10^-23;
T=300;
N0=k*T;
Noise_power=N0/2;

Eb_N0_dB=zeros(1,((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1);
Eb_N0=zeros(1,((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1);

BER_AWGN2ASK_sim=zeros(1,((Eb_N0_dB_Stop-
Eb_N0_dB_Start)/2)+1);
BER_AWGN2ASK_sim4=zeros(1,((Eb_N0_dB_Stop-
Eb_N0_dB_Start)/2)+1);
BER_AWGN2ASK_sim8=zeros(1,((Eb_N0_dB_Stop-
Eb_N0_dB_Start)/2)+1);

for i=1:1:((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1
Eb_N0_dB(i)=Eb_N0_dB_Start+2*(i-1);
Eb_N0(i)=10^(Eb_N0_dB(i)/10);
Eb=Eb_N0(i)*N0;

Transmitted_symbolsN2=sqrt(Eb).*askmodulation(bits, 2);
Transmitted_symbolsN4=sqrt((2*Eb)/5).*askmodulation(bits, 
4);
Transmitted_symbolsN8=sqrt((3*Eb)/21).*askmodulation(bits, 
8);

Gaussian_noise2=0+sqrt(Noise_power).*randn(length(Transmitt
ed_symbolsN2), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsN2), 
1);
Gaussian_noise4=0+sqrt(Noise_power).*randn(length(Transmitt
ed_symbolsN4), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsN4), 
1);
```

```
Gaussian_noise8=0+sqrt(Noise_power).*randn(length(Transmitt
ed_symbolsN8), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsN8),
1);

Received_symbolsN2=Transmitted_symbolsN2+Gaussian_noise2;
Received_symbolsN4=Transmitted_symbolsN4+Gaussian_noise4;
Received_symbolsN8=Transmitted_symbolsN8+Gaussian_noise8;

Normalized_symbolsN2=Received_symbolsN2./sqrt(Eb);
Normalized_symbolsN4=Received_symbolsN4./sqrt((2*Eb)/5);
Normalized_symbolsN8=Received_symbolsN8./sqrt((3*Eb)/21);

Received_bitsN2=askdemodulation(Normalized_symbolsN2, 2);
Received_bitsN4=askdemodulation(Normalized_symbolsN4, 4);
Received_bitsN8=askdemodulation(Normalized_symbolsN8, 8);

BER_AWGN2ASK_sim(i)=sum(bits~=Received_bitsN2)/length(bits)
;
BER_AWGN2ASK_sim4(i)=sum(bits~=Received_bitsN4)/length(bits
);
BER_AWGN2ASK_sim8(i)=sum(bits~=Received_bitsN8)/length(bits
);
end

semilogy(Eb_N0_dB,BER_AWGN2ASK_sim,'-g');
hold on
semilogy(Eb_N0_dB,BER_AWGN2ASK_sim4,'-k');
hold on
semilogy(Eb_N0_dB,BER_AWGN2ASK_sim8,'-m');
xlabel('Eb/N0 in dB');
ylabel('BER');
end
```

Task 4 Source code

```
clc; %clear the command window
clear all;  %clear workspace

%%

ImageA=imread('lena256.jpg');
[len,wid,hei]=size(ImageA);
bits_per_value=8;
fprintf(' The length : %d\n The width  : %d\n The height : %d\n', len, wid, hei);
fprintf('(The length and the width are defined by the image pixels and height is 3 for RGB images)\n');

figure
image(ImageA);
title('Imported original image');
axis off;

fprintf('Imported  image  \n .. \n .. \n .. ');

fprintf('Displaying Imported Image. \nProgram paused. Press enter to Excercise 2.\n');
pause;

%%

fprintf('\n Wait! It might take some time \n Image to bits conversion in progress... \n');
bits=image2bits(ImageA, len, wid, hei, bits_per_value);
fprintf('\n First 100 bits: ');
fprintf('%d', bits(1:100));
fprintf('\n');
fprintf('\nProgram paused. Press enter to Excercise 3.\n');
pause;


%%

figure
plot_bertoaskT4(-4, 10, bits);
```

```
legend('2-ASK Simulation','2-ASK with convolution code rate
1/2');
xlabel('Eb/N0 in dB');
ylabel('BER');
```

"plot_bertoask_t4.m" function Source code

```
function plot_bertoask_t4(Eb_N0_dB_Start, Eb_N0_dB_Stop,
bits)

k=1.38*10^-23;
T=300;
N0=k*T;
Noise_power=N0/2;

coded_bits_1_2=channel_coder(bits,'conv_R=1_2');

Eb_N0_dB=zeros(1,((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1);
Eb_N0=zeros(1,((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1);

BER_AWGN2ASK_sim=zeros(1,((Eb_N0_dB_Stop-
Eb_N0_dB_Start)/2)+1);
BER_AWGN2ASK_1_2_sim=zeros(1,((Eb_N0_dB_Stop-
Eb_N0_dB_Start)/2)+1);

for i=1:1:((Eb_N0_dB_Stop-Eb_N0_dB_Start)/2)+1
Eb_N0_dB(i)=Eb_N0_dB_Start+2*(i-1);
Eb_N0(i)=10^(Eb_N0_dB(i)/10);
Eb=Eb_N0(i)*N0;

Transmitted_symbolsN=sqrt(Eb).*askmodulation(bits, 2);
Transmitted_symbolsN_1_2=sqrt(0.5*Eb).*askmodulation(coded_
bits_1_2, 2);

Gaussian_noise=0+sqrt(Noise_power).*randn(length(Transmitte
d_symbolsN), 1)...

+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsN),
1);
Gaussian_noise_1_2=0+sqrt(Noise_power).*randn(length(Transm
itted_symbolsN_1_2), 1)...
```

```
+1i*sqrt(Noise_power).*randn(length(Transmitted_symbolsN_1_
2), 1);

Received_symbolsN=Transmitted_symbolsN+Gaussian_noise;
Received_symbolsN_1_2=Transmitted_symbolsN_1_2+Gaussian_noi
se_1_2;

Normalized_symbolsN=Received_symbolsN./sqrt(Eb);
Normalized_symbolsN_1_2=Received_symbolsN_1_2./sqrt(0.5*Eb)
;

Received_bitsN=askdemodulation(Normalized_symbolsN, 2);
Received_bitsN_1_2=askdemodulation(Normalized_symbolsN_1_2,
2);

BER_AWGN2ASK_sim(i)=sum(bits~=Received_bitsN)/length(bits);
BER_AWGN2ASK_1_2_sim(i)=sum(coded_bits_1_2~=Received_bitsN_
1_2)/length(coded_bits_1_2);
end

semilogy(Eb_N0_dB,BER_AWGN2ASK_sim,'--.k');
hold on
semilogy(Eb_N0_dB,BER_AWGN2ASK_1_2_sim,'-k');
xlabel('Eb/N0 in dB');
ylabel('BER');
end
```