# ECE 463 Lab 8: Modulation III : QAM, PSK, BER vs SNR

## 1. Introduction

So far, we have focused mainly on binary modulation using simple forms of ASK, FSK, and PSK. In this lab, we will increase the number of bits encoded in one symbol and generalize the encoding and decoding for multi-symbols. Given the AWGN channel and the equal probability of the transmitted symbols, the maximum likelihood decision rule is simply to choose the symbols closest to the received symbols. Under this assumption, we are able to derive BER with respect to SNR. We will verify the relationship by experiments and compare the results with the theory.

### 1.1. Contents

1. Introduction
2. Symbol Encoding and Decoding
3. M-ary Modulations
4. AWGN
5. Questions

### 1.2. Report

Submit the answers, figures and the discussions on all the questions. The report is due as a hard copy at the beginning of the next lab.

## 2. Symbol Encoding and Decoding

So far, we simply mapped the input bit streams bit-by-bit into the symbol '1' and '-1' for BPSK. In this section, you will implement a generalized symbol encoder/decoder.

- Symbol_encoder.gvi
- Symbol_decoder.gvi

### 2.1. Symbol Encoder (bit-to-symbol)

We will implement a symbol encoder that translates the input bit stream into the desired symbols. The symbol map array contains the mapping information.

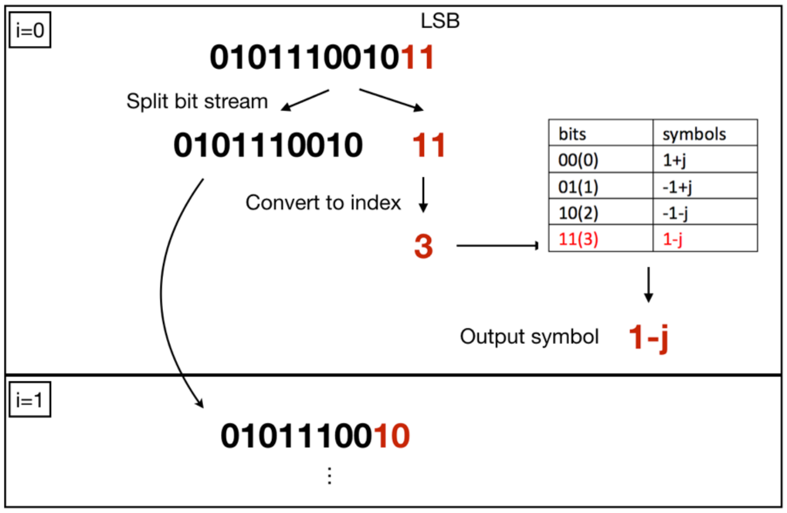
- If N bits are encoded into a symbol, $\boldsymbol{M = 2^N}$ symbols are required.

- For example, a possible QPSK symbol mapping can be

| bits | symbols |
|---|---|
| 00(0) | 1+j |
| 01(1) | -1+j |
| 10(2) | -1-j |
| 11(3) | 1-j |

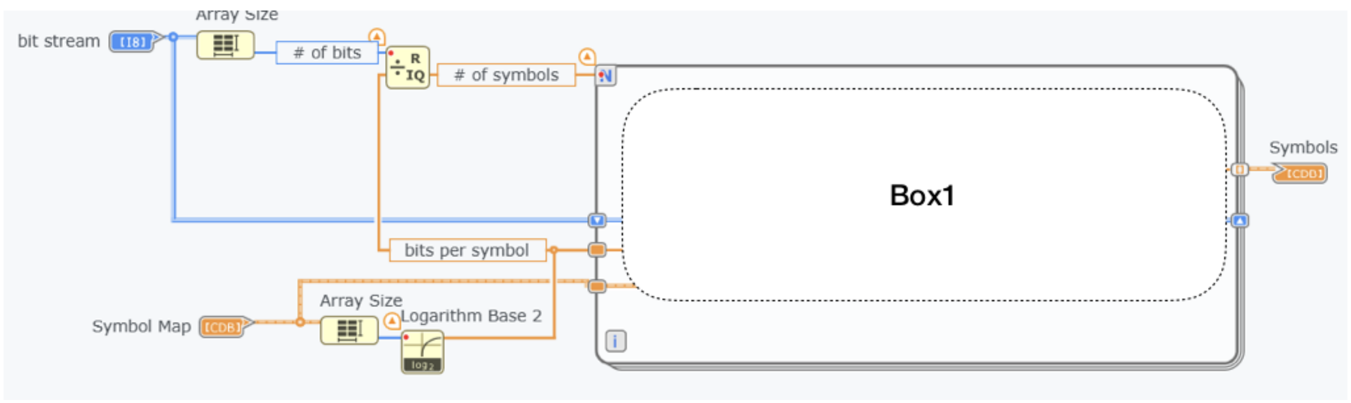
In this case, the symbol map array is [1+j, -1+j, -1-j, 1-j].

- The symbol encoder works as follows. First, it divides the input bit stream into two: the first $\boldsymbol{\log_2 M}$ bits and the rest of bits. Then, the first portion is converted into the decimal number (index) and the symbol corresponding to the index is returned. The rest bit stream is used for the input in the next iteration. See the diagram below for QPSK example.



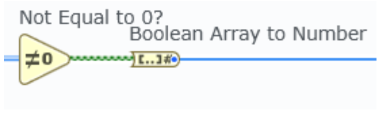
Complete the figure below to implement "Symbol_encoder.gvi".

| **Symbol_encoder.gvi** | Terminal name | Type | Description |
|---|---|---|---|
| Input | *Bit stream* | Singed integer array (1D) | |
| | *Symbol map* | Complex double array (1D) | |
| Output | *Symbols* | Complex double array (1D) | |



- Tip1: Use “split 1D array” block to divide an array into two portions.

- Tip2: Use the following chains to convert a portion of bit stream into the index for symbol map.



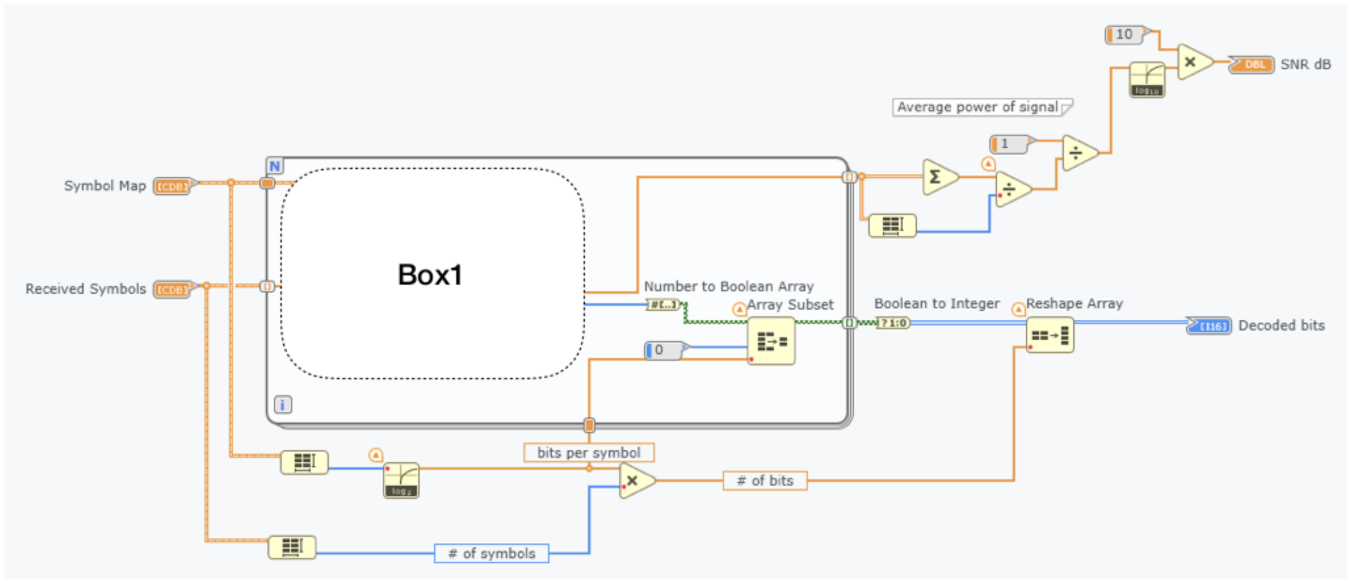
## 2.2. Symbol Decoder (symbol-to-bit)

We will implement a symbol decoder that performs the maximum likelihood decoding and translates the received symbols into the bit stream. The decoder shares the same symbol map with the encoder.

- Given the AWGN channel and the equal probability of the transmitted symbols, the maximum likelihood decision rule is simply to choose the symbols closest to the received symbols.

- Complete Box1 in the figure. The box 1) calculates the squared Euclidean distance between the received symbol and the M transmitted symbols, 2) finds the index for the symbol with the minimum squared distance, 3) finds the minimum squared distance. The SNR of the signal will be calculated by the calculated distance.

| **Symbol_decoder.gvi** | Terminal name | Type | Description |
|---|---|---|---|
| Input | *Received symbols* | Complex double array (1D) | |
| | *Symbol map* | Complex double array (1D) | |
| Output | *Decoded bits* | Signed integer array (1D) | |
| | *SNR (dB)* | Double | |



## 2.3.Simulation

Verify the created encoder and decoder by simulation. Try a simple modulation (e.g. QPSK). Generate a short bit stream to see the encoder and decoder works.

## 3. M-ary Modulations

### 3.1. MPSK

In this section, we will modify the old BPSK diagram and implement MPSK. First, create “gen_PSKmap.gvi” that generates a symbol map for MPSK. The symbols are defined as

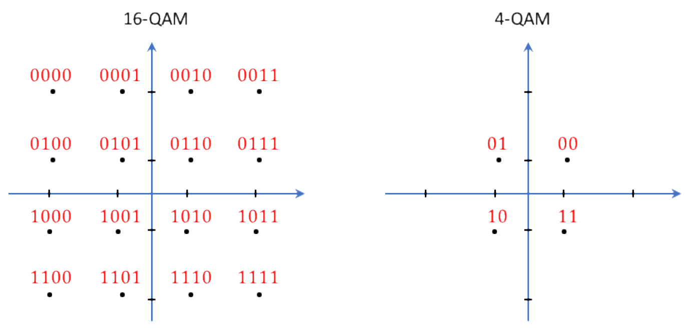
$$s_k = e^{\frac{j2\pi k}{M}} \quad , k = 0, \ldots, M-1$$

| **gen_PSKmap.gvi** | Terminal name | Type | Description |
|---|---|---|---|
| Input | *M (symbol levels)* | Double | |
| Output | *Symbol map* | Complex double array (1D) | |

- In Tx, use “Symbol_encoder.gvi” and “gen_PSKmap.gvi” to replace BPSK encoding.
- In Rx, use “Symbol_decoder.gvi” and “gen_PSKmap.gvi” to replace BPSK decoding.

## 3.2. MQAM

In this section, we will implement 4-QAM and 16-QAM. You can either implement a generalized QAM map using nested for loops or simply use a constant array. The symbol maps for 4-QAM and 16-QAM are defined as the below figure.



- Use your QAM symbol maps to replace the MPSK symbol map.
- In Tx, use the right factor to normalize the transmit power. The average power should be 1.
- You do not need to change the training sequence.
- In Rx, remove all the normalization blocks and let the channel correction do the normalization. The old normalization block will not work properly because the power of QAM signal is not uniformly 1.
- Since the normalization is removed before the frame sync, it is better to use the normalized correlation to detect the training sequence. This ensures the peak correlation is around 1. Use the threshold close to 1.

$$R[n] = \frac{\left|\sum_{k=0}^{N_t-1} y^*[n+k]y[n+k+N_t]\right|}{\sqrt{\sum_{k=0}^{N_t-1} |y[n+k]|^2}\sqrt{\sum_{k=0}^{N_t-1} |y[n+k+N]|^2}}$$

## 4. Additive White Gaussian Noise (AWGN)

To manipulate the SNR, we will generate AWGN in the received symbols. For the sake of convenience, the noise will be injected **after the frame sync, CFO correction and channel estimation**. Create “Add_AWGN.gvi” that adds Gaussian noise in the received symbols.

| **Add_AWGN.gvi** | Terminal name | Type | Description |
|---|---|---|---|
| Input | *Received symbols* | Complex double array (1D) | |
| | *Noise standard deviation* | Double | |
| Output | *Output symbols* | Complex double array (1D) | |

- Use “Gaussian White Noise” block to generate Gaussian noise. Note that the output of the block is double-type. Split the received symbols in real and imaginary part, then add the noise.

## 5. Questions

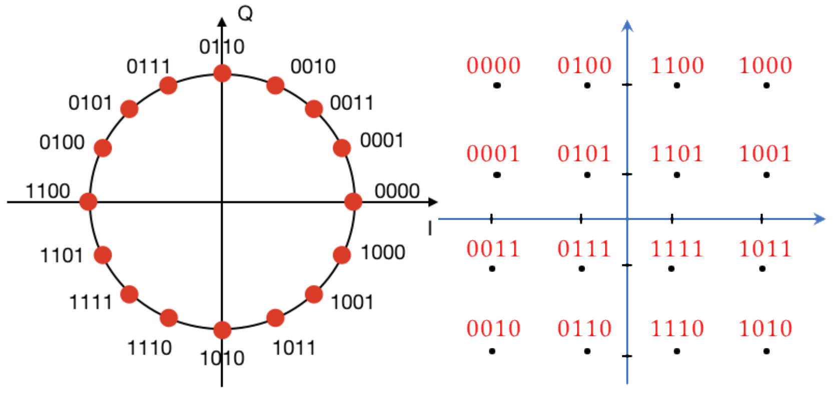
5.1.1. (USRP) Plot the measured constellation of

- 4-PSK
- 8-PSK
- 16-PSK
- 4-QAM
- 16-QAM

Do not generate AWGN noise for this question. Use a 30dB attenuator.

5.1.2. (USRP) Use Gray codes for the following modulations.

- 16-PSK
- 16-QAM



Increase the AWGN in Rx and measure the SNR and BER (You may increase the message length to get more precise BER.). Compare the results with the non-Gray coded symbol map. What difference did you measure? Compare with the theoretical value.

5.1.3. (USRP) Your task is to obtain the BER vs Eb/N0 curve for the following modulations.

- BPSK
- 4-QAM
- 16-QAM
- 64-QAM (extra credit)

First, plot the theoretical curves using the equations in the lecture note. Then, measure the BER by changing the SNR to plot the measured curves.

5.1.4. In the lecture note, the maximum likelihood decoder was derived by assuming the probability of transmitting bit 1 and 0 are the same. Assume the probability of transmitting 1 is $\frac{3}{4}$. What is the new maximum likelihood decoder?