

Simulating Different Receivers in a Rayleigh Fading, SISO Environment

Project #1

Intelligent Communication Systems (ICS) Lab.
노용재

Winter Intern Seminar (2023-1)

Contents

1	Implementation	1
1.1	Common Environment Variables	1
1.2	ZF(Zero-forcing)	2
1.3	MMSE(Minimum Mean Square Error)	2
1.4	MLD(Maximum Likelihood Detection)	3
2	결과 및 분석	3
2.1	Simulation Result	3
2.2	Normalization Factor	3
2.3	Binary Modulation, 4-QAM에서 Receiver간 BER의 동일성	7
3	미해결 과제 外 의문점 및 질문	8
4	Entire Code	8

Binary Modulation, 4-QAM, 16-QAM의 modulation 환경에서 Zero-forcing (ZF) receiver, Minimum Mean Square Error (MMSE) receiver, Maximum Likelihood Detection (MLD)의 BER (bit error rate)를 구하기 위해 실험을 MATLAB에서 수행하였다. 실험의 공통된 조건은 다음과 같다.

- E_s/N_0 는 -2dB 20dB(2dB 간격)
- 신호의 평균전력은 1W이 되도록한다. ($E_s = 1$)
- SISO; Single Input, Single Output

1 Implementation

$M = 2^{2n}$ ($n = 1, 2, 3, \dots$)의 상황을 가정하였다.

1.1 Common Environment Variables

$$y = hs + n \quad (1)$$

```

1 EsNO_dB = -2:2:20;
2 EsNO = db2pow(EsNO_dB);
3
4 EbNO = EsNO / log2(M);
5 EbNO_dB = pow2db(EbNO);
6
7 NumberOfSignals = 1;
8 LengthBitSequence = NumberOfSignals*log2(M); % log2(M) bits per signal
9
10 % Bit Generation
11 BitSequence = randi([0 1], 1, LengthBitSequence);
12 SymbolSequence = qammod(BitSequence.', M, 'InputType', 'bit', '
    UnitAveragePower', 1).';
13
14 % Noise (n) Generation
15 NoiseSequence = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
    SymbolSequence))) / sqrt(2);
16
17 % Channel (h) Generation
18 H = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
    SymbolSequence))) ./ sqrt(2);
19
20 % Received Signal (y = s + n) Generation
21 ReceivedSymbolSequence = H .* SymbolSequence + NoiseSequence * sqrt(1 /
    EsNO(indx_EbNO));

```

1.2 ZF(Zero-forcing)

$$z = wy : w_{ZF} = (h)^{-1} \quad (2)$$

$$\hat{s} = \underset{s}{\operatorname{argmin}} |z - s|^2 \quad (3)$$

```

1 w_zf = H.^(-1);
2 DetectionSymbolSequence_ZF = ReceivedSymbolSequence .* w_zf; % Detection (
    Zero-Forcing: y / h)
3
4 DetectionBitSequence_ZF = qamdemod(DetectionSymbolSequence_ZF.', M, '
    OutputType', 'bit', 'UnitAveragePower', 1)';

```

1.3 MMSE(Minimum Mean Square Error)

$$z = wy : w_{MMSE} = (|h|^2 + 1/\rho)^{-1} h^* \quad (4)$$

$$\hat{s} = \underset{s}{\operatorname{argmin}} |z - s|^2 \quad (5)$$

```

1 w_mmse = (abs(H).^2+1/EsNO(indx_EbNO)).^(-1) .* conj(H);
2 DetectionSymbolSequence_MMSE = ReceivedSymbolSequence .* w_mmse;
3 DetectionBitSequence_MMSE = qamdemod(DetectionSymbolSequence_MMSE.', M, '
    OutputType', 'bit', 'UnitAveragePower', 1)';

```

1.4 MLD(Maximum Likelihood Detection)

$$\hat{s} = \underset{s}{\operatorname{argmin}} |y - hs|^2 \quad (6)$$

```

1 alphabet = qammod([0:M-1], M, 'UnitAveragePower', true);
2 arg = (ones(length(alphabet),1) * ReceivedSymbolSequence) - (alphabet.' *
  H);
3 [val,idx] = min(abs(arg).^2);
4 DetectionBitSequence_MLD = reshape(de2bi(idx-1, log2(M), 'left-msb'),' , 1,
  []);

```

arg 는 $(y - hs)$ 를 나타낸 matrix이다. arg 의 $length(alphabet) \times length(SymbolSequence)$ 이다. $(M \times l)$

$$ReceivedSymbolSequence = \{d_1, d_2, \dots, d_l\} \quad (7)$$

$$alphabet = \{a_1, a_2, \dots, a_M\} \quad (8)$$

$$H = \{h_1, h_2, \dots, h_l\} \quad (9)$$

$$(10)$$

$$\begin{aligned}
 arg &= \begin{bmatrix} d_1 & d_2 & \dots & d_{l-1} & d_l \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_1 & d_2 & \dots & d_{l-1} & d_l \end{bmatrix} - \begin{bmatrix} a_1 h_1 & a_1 h_2 & \dots & a_1 h_{l-1} & a_1 h_l \\ a_2 h_1 & a_2 h_2 & \dots & a_2 h_{l-1} & a_2 h_l \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{M-1} h_1 & a_{M-1} h_2 & \dots & a_{M-1} h_{l-1} & a_{M-1} h_l \\ a_M h_1 & a_M h_2 & \dots & a_M h_{l-1} & a_M h_l \end{bmatrix} \\
 &= \begin{bmatrix} d_1 - a_1 h_1 & d_2 - a_1 h_2 & \dots & d_{l-1} - a_1 h_{l-1} & d_l - a_1 h_l \\ d_1 - a_2 h_1 & d_2 - a_2 h_2 & \dots & d_{l-1} - a_2 h_{l-1} & d_l - a_2 h_l \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_1 - a_{M-1} h_1 & d_2 - a_{M-1} h_2 & \dots & d_{l-1} - a_{M-1} h_{l-1} & d_l - a_{M-1} h_l \\ d_1 - a_M h_1 & d_2 - a_M h_2 & \dots & d_{l-1} - a_M h_{l-1} & d_l - a_M h_l \end{bmatrix} \quad (11)
 \end{aligned}$$

2 결과 및 분석

2.1 Simulation Result

이론값은 Matlab의 *berfading* 함수를 통해 얻어졌다.

2.2 Normalization Factor

코드 내에서 직접적으로 쓰이지는 않았지만 생각해볼 만한 부분은 *Normalization Factor*이다. 이 *Normalization Factor*를 사용하여 평균 전력이 1W가 되게끔 할 수 있다.

Code1

```
alphabet = qammod([0:M-1], M, 'UnitAveragePower', true);
```

Code 2

```
Normalization_Factor = sqrt(2/3*(M-1));
alphabet = qammod([0:M-1], M) / Normalization_Factor;
```

Code 1과 Code 2는 동일한 결과를 이룬다.

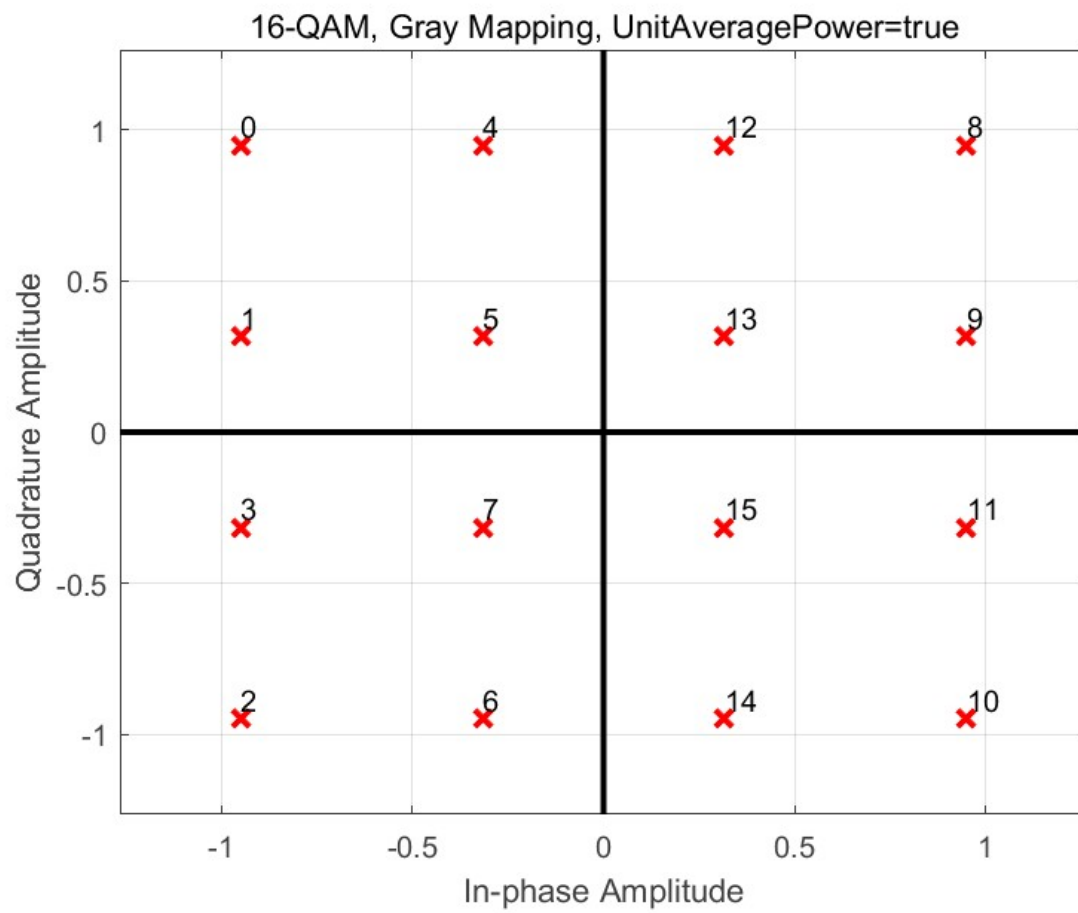


Figure 1: Binary Modulation

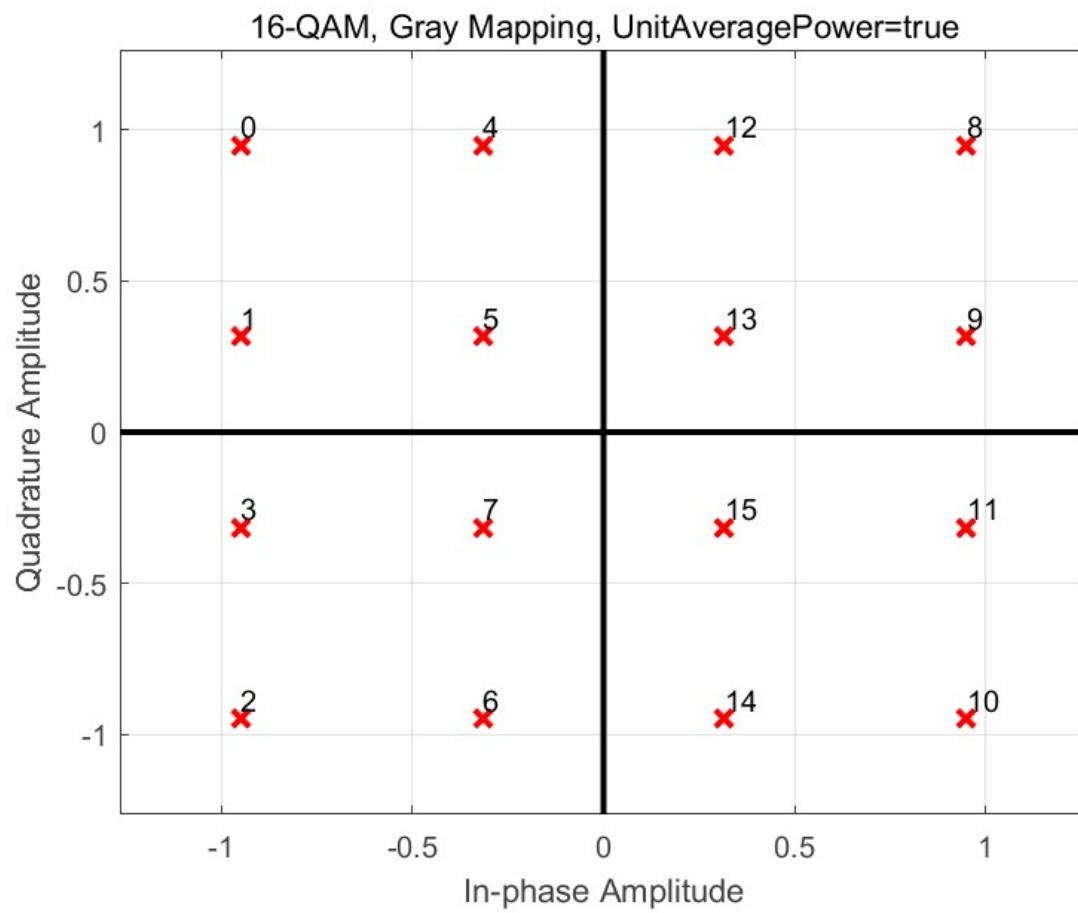


Figure 2: 4-QAM

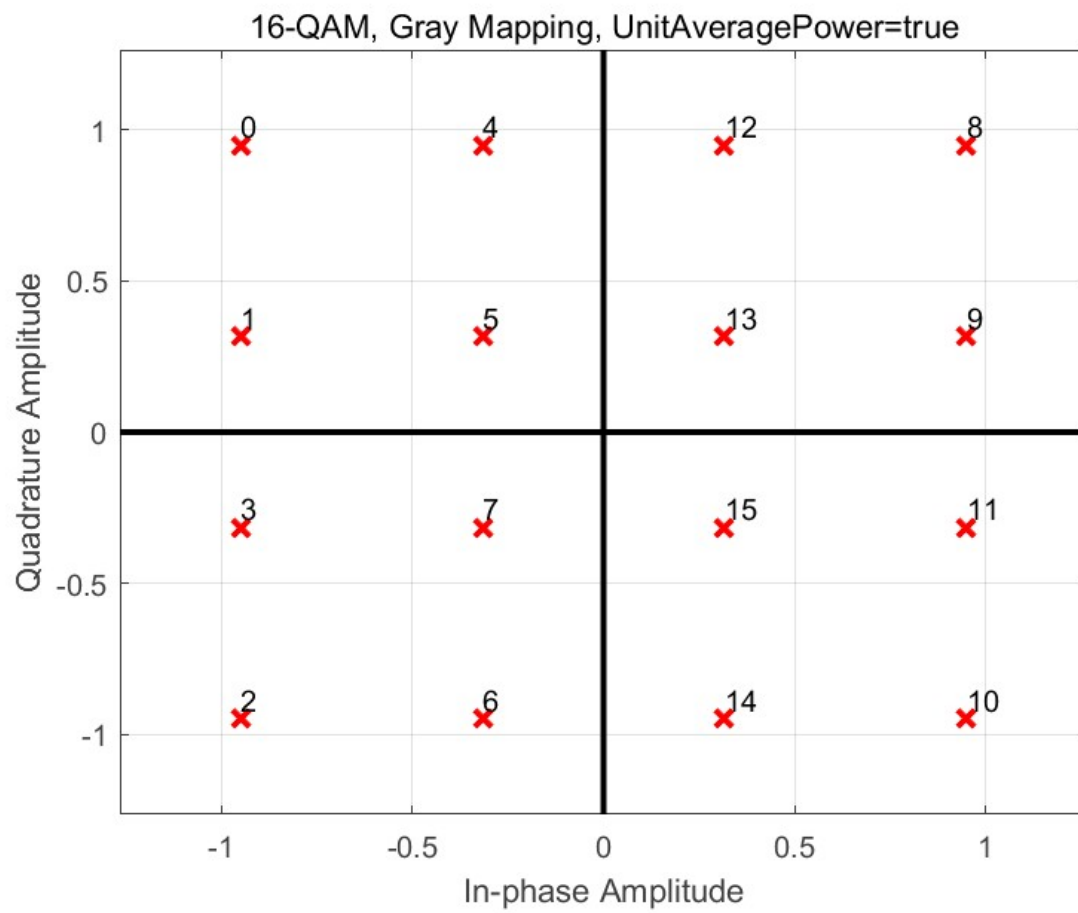


Figure 3: 16-QAM

$M = 2^{2n}$ ($n = 1, 2, 3, \dots$)일 때의 *Normalization Factor*를 일반화 시켜보겠다.
일반적인 QAM의 Constellation Diagram을 살펴보면 실수 \sqrt{M} 개, 허수 \sqrt{M} 개의 point를 갖는 것을 알 수 있다.
하나의 신호에 대한 값을 그 신호의 *alphabet*이라고하자. M 개의 *alphabet*이 다음과 같다고하자.

$$alphabet = \pm(2n-1) \pm j \cdot (2n-1) \quad n \in 1, 2, \dots, \frac{\sqrt{M}}{2} \quad (12)$$

그렇다면 신호의 평균전력은 다음과 같이 일반화 가능하다.

$$\begin{aligned} E_s = E[|s|^2] &= \frac{1}{M} \sum_{n=1}^M |s_n|^2 \\ &= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2 + (2m-1)^2] \\ &= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2] + \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2m-1)^2] \\ &= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2] \cdot 2 \\ &= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2 \cdot \sqrt{M}] \\ &= \frac{4}{\sqrt{M}} \sum_{n=1}^{\frac{\sqrt{M}}{2}} [4n^2 - 4n + 1] \\ &= \frac{2}{3}(M-1) \end{aligned} \quad (13)$$

(13)에서의 결과를 토대로 normalization이 이뤄진 alphabet을 구할 수 있다.

$$normalized\ alphabet = \left[\pm \frac{2n-1}{\sqrt{\frac{2}{3}(M-1)}} \pm j \cdot \frac{2n-1}{\sqrt{\frac{2}{3}(M-1)}} \right] \quad n \in \{1, 2, \dots, \sqrt{M}\} \quad (14)$$

해당 결과를 토대로 다시 평균 전력을 구한다면 E_s 가 1W임을 확인할 수 있다.

참고자료

다음은 16-QAM의 Constellation이다.

2.3 Binary Modulation, 4-QAM에서 Receiver간 BER의 동일성

실험 결과에 따르면, Binary Modulation일 때와 4-QAM일 때 어떤 ZF, MMSE, MLD 방식을 사용한지와 무관하게 BER이 동일한 것을 관찰할 수 있었다.

ZF와 MMSE의 경우를 먼저 살펴보자. ZF와 MMSE에 해당하는 조건식들은 다음과 같다.

$$\hat{s} = \underset{s}{\operatorname{argmin}} |z - s|^2 \quad (15)$$

$$z = w(hs + n) \quad (16)$$

$$w_{ZF} = (h)^{-1} \quad (17)$$

$$w_{MMSE} = (|h|^2 + 1/\rho)^{-1} h^* \quad (18)$$

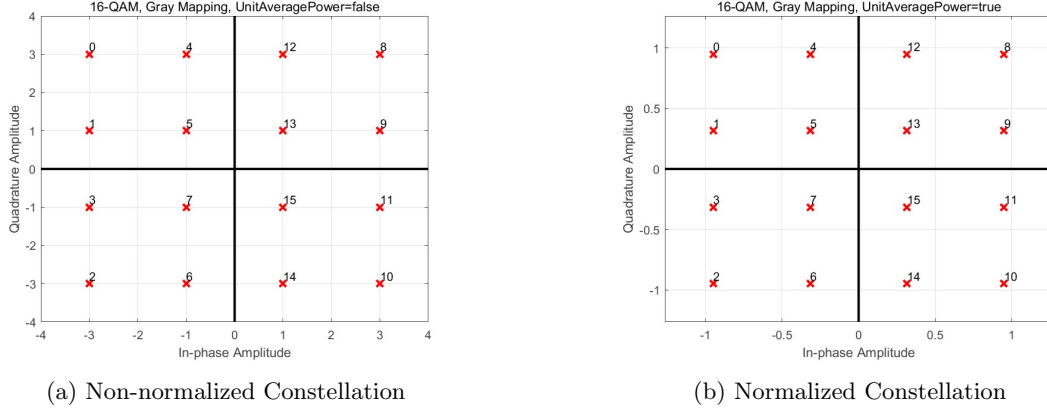


Figure 4: 16-QAM Constellation

z 는 post-processing signal에 해당된다. ZF와 MMSE, 각각의 post-processing signal을 살펴보겠다.

ZF의 Post-processing Signal

$$\begin{aligned}
 z_{ZF} &= w_{ZF}(hs + n) \\
 &= h(-1)(hs + n) \\
 &= s + \frac{n}{h}
 \end{aligned} \tag{19}$$

MMSE의 Post-processing Signal

$$\begin{aligned}
 z_{MMSE} &= w_{MMSE}(hs + n) \\
 &= (|h|^2 + 1/\rho)^{-1}h^*(hs + n) \\
 &= (|h|^2 + 1/\rho)^{-1}h^*h(s + \frac{n}{h})
 \end{aligned} \tag{20}$$

위의 결과를 토대로 다음과 같이 표현이 가능하다.

$$\begin{aligned}
 z_{MMSE} &= (|h|^2 + 1/\rho)^{-1}h^*h \cdot z_{ZF} \\
 \text{sgn}(z_{MMSE}) &= \text{sgn}((|h|^2 + 1/\rho)^{-1}h^*h \cdot z_{ZF}) \\
 &= \text{sgn}(z_{ZF}) \quad (\because (|h|^2 + 1/\rho)^{-1}h^*h \geq 0)
 \end{aligned} \tag{21}$$

$\text{sgn}(z_{MMSE}) = \text{sgn}(z_{ZF})$ 라는 것은 극 좌표계로 나타냈을 때의 각도가 같다는 것을 의미한다. **Binary modulation**이나 **4-QAM**의 경우 **amplitude**가 아닌 **phase**에만 영향을 받기 때문에 감지되는 signal은 receiver가 ZF인지, MMSE인지에 관계없이 동일해진다.

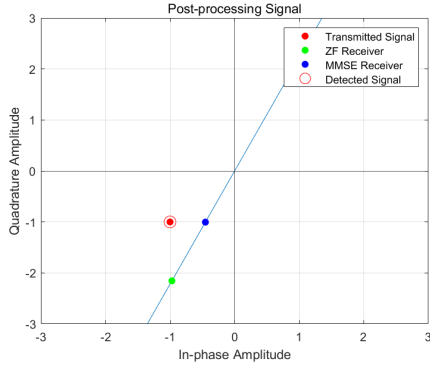
Figure 3는 *Matlab*을 통해 z_{MMSE} 와 z_{ZF} 를 직교좌표계로 바꾼 뒤, 두 θ 값을 비교한 것이다. θ 값의 차이 중 가장 큰 값은 $4.4409e^{-16}$ 이었다. 이는 매우 작은 값으로 컴퓨터가 가지는 'finite precision'로 인해 생겨난 오차로 생각할 수 있다. 그러므로 모든 경우에 대해서 $\theta_{MMSE} - \theta_{ZF} = 0$ 가 나타났다고 생각할 수 있다. **Figure 2**의 ZF Receiver와 MMSE Receiver가 하나의 직선 위에 나타난 것을 통해 이를 시각적으로도 확인할 수 있다.

3 미해결 과제 外 의문점 및 질문

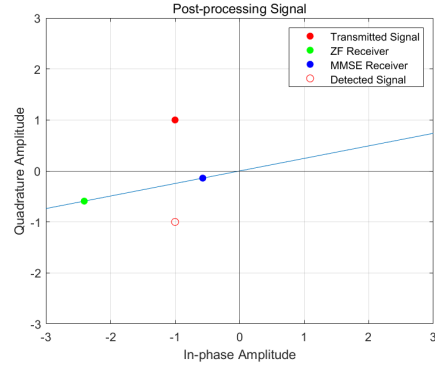
- 왜 SNR이 커질 때도 BER이 다른가??

4 Entire Code ¹

¹Also uploaded to <https://github.com/lightwick/ICS-project>



(a) Correct Detection Example



(b) Signal Detection Error Example

Figure 5: 4-QAM의 ZF, MMSE Post-processing Signal Example

```
>> [theta_zf, rho_zf] = cart2pol(real(DetectionSymbolSequence_ZF), imag(DetectionSymbolSequence_ZF));
>> [theta_mmse, rho_mmse] = cart2pol(real(DetectionSymbolSequence_MMSE), imag(DetectionSymbolSequence_MMSE));
>> max(abs(theta_zf - theta_mmse))

ans =

4.4409e-16
```

Figure 6: EsN0=5dB

```
1 close all
2 clear
3 clc
4
5 % Simulation
6 M = 16
7 Nt = 1;
8 NumberOfSignals = 10^2;
9 LengthBitSequence = Nt * NumberOfSignals*log2(M); % log2(M) bits per
   signal
10
11 NumberIteration = 10^3;
12
13 Es = 1;
14
15 EsN0_dB = -2:2:20;
16 EsN0 = db2pow(EsN0_dB);
17
18 EbN0 = EsN0 / log2(M);
19 EbN0_dB = pow2db(EbN0);
20
21 ErrorCount_ZF = zeros(1, length(EbN0_dB));
22 ErrorCount_MMSE = zeros(1, length(EbN0_dB));
23 ErrorCount_MLD = zeros(1, length(EbN0_dB));
24
25 alphabet = qammod([0:M-1], M, 'UnitAveragePower', true);
```

```

26
27 for iTTotal = 1 : NumberIteration
28     BitSequence = randi([0 1], 1, LengthBitSequence); % Bit Generation (
29         BitSequence = rand(1, LengthBitSequence) > 0.5;)
30     SymbolSequence = qammod(BitSequence.', M, 'InputType', 'bit', '
31         UnitAveragePower', 1).';
32     %avgPower = mean(abs(SymbolSequence).^2)
33     NoiseSequence = (randn(1, length(SymbolSequence)) + 1j * randn(1,
34         length(SymbolSequence))) / sqrt(2); % Noise (n) Generation
35     H = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
36         SymbolSequence))) ./ sqrt(2); % Channel (h) Generation
37     for indx_EbN0 = 1 : length(EbN0)
38         ReceivedSymbolSequence = H .* SymbolSequence + NoiseSequence *
39             sqrt(1 / EsN0(indx_EbN0)); % Received Signal (y = s + n)
40             Generation
41
42         % ZF Receiver
43         w_zf = H.^(-1);
44         DetectionSymbolSequence_ZF = ReceivedSymbolSequence .* w_zf; %
45             Detection (Zero-Forcing: y / h)
46
47         % MMSE Receiver
48         w_mmse = (abs(H).^2+1/EsN0(indx_EbN0)).^(-1) .* conj(H);
49         DetectionSymbolSequence_MMSE = ReceivedSymbolSequence .* w_mmse;;
50
51         % MLD Receiver;
52         arg = (ones(length(alphabet),1) * ReceivedSymbolSequence) - (
53             alphabet.' * H);
54         arg = abs(arg).^2;
55         [val,idx] = min(arg);
56
57         % Symbol Sequence -> Bit Sequence
58         DetectionBitSequence_ZF = qamdemod(DetectionSymbolSequence_ZF.', M
59             , 'OutputType', 'bit', 'UnitAveragePower', 1)'; % Detection
60         DetectionBitSequence_MMSE = qamdemod(DetectionSymbolSequence_MMSE
61             .', M, 'OutputType', 'bit', 'UnitAveragePower', 1)'; % tmp
62             value;
63         DetectionBitSequence_MLD =
64
65         ErrorCount_ZF(1, indx_EbN0) = ErrorCount_ZF(1, indx_EbN0) + sum(
66             DetectionBitSequence_ZF~=BitSequence);
67         ErrorCount_MMSE(1, indx_EbN0) = ErrorCount_MMSE(1, indx_EbN0) +
68             sum(DetectionBitSequence_MMSE~=BitSequence);
69         ErrorCount_MLD(1, indx_EbN0) = ErrorCount_MLD(1, indx_EbN0) + sum(
70             DetectionBitSequence_MLD~=BitSequence);
71
72     end
73 %     toc
74
75 end
76
77 BER_Simulation_ZF = ErrorCount_ZF / (LengthBitSequence * NumberIteration);
78 BER_Simulation_MMSE = ErrorCount_MMSE / (LengthBitSequence *
79     NumberIteration);

```

```

64 BER_Simulation_MLD = ErrorCount_MLD / (LengthBitSequence * NumberIteration
    );
65
66 if M==2
67     BER_Theory = berfading(EbNO_dB, 'psk', 2, 1);
68 else
69     BER_Theory = berfading(EbNO_dB, 'qam', M, 1); % not sure if 'dataenc'
        needs to be specified; I don't even know what it does
70 end
71
72 % Plot
73 figure()
74 semilogy(EsNO_dB, BER_Theory, 'r--');
75 hold on
76 semilogy(EsNO_dB, BER_Simulation_ZF, 'bo');
77 semilogy(EsNO_dB, BER_Simulation_MMSE, 'bx');
78 semilogy(EsNO_dB, BER_Simulation_MLD, 'b^');
79
80
81 axis([-2 20 10^-3 0.5])
82 grid on
83 legend('Theory (Rayleigh)', 'ZF (Rayleigh)', 'MMSE (Rayleigh)', 'MLD (
    Rayleigh)');
84 xlabel('Es/No [dB]');
85 ylabel('BER');
86 title('BER for QAM (M='+string(M)+'')');

```