Simulating Different Receivers in a Rayleigh Fading, SISO Environment

Project #1

Intelligent Communication Systems (ICS) Lab.
노용재

Winter Intern Seminar (2023-1)

Contents

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.

- Es/N0는 -2dB 20dB(2dB 간격)
- 신호의 평균전력은 1W이 되도록한다. ($Es = 1$)
- SISO; Single Input, Single Output

1 Implementation

$M = 2^2n$ ($n = 1, 2, 3, ...$)의 상황을 가정하였다.

1.1 Common Environment Variables

$$y = hs + n \tag{1}$$

```
EsN0_dB = -2:2:20;
EsN0 = db2pow(EsN0_dB);

EbN0 = EsN0 / log2(M);
```

```
EbN0_dB = pow2db(EbN0);

LengthBitSequence = NumberOfSignals*log2(M); % log2(M) bits per signal

% Bit Generation
BitSequence = randi([0 1], 1, LengthBitSequence);
SymbolSequence = qammod(BitSequence.', M, 'InputType', 'bit', '
    UnitAveragePower', 1).';

% Noise (n) Generation
NoiseSequence = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
    SymbolSequence))) / sqrt(2);

% Channel (h) Generation
H = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
    SymbolSequence))) ./ sqrt(2);

% Received Signal (y = s + n) Generation
ReceivedSymbolSequence = H .* SymbolSequence + NoiseSequence * sqrt(1 /
    EsN0(indx_EbN0));
```

1.2 ZF(Zero-forcing)

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

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

```
w_zf = H.^(-1);
DetectionSymbolSequence_ZF = ReceivedSymbolSequence .* w_zf; % Detection (
    Zero-Forcing: y / h)

DetectionBitSequence_ZF = qamdemod(DetectionSymbolSequence_ZF.', M, '
    OutputType', 'bit', 'UnitAveragePower', 1)';
```

1.3 MMSE(Minimum Mean Square Error)

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

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

```
w_mmse = (abs(H).^2+1/EsN0(indx_EbN0)).^(-1) .* conj(H);
DetectionSymbolSequence_MMSE = ReceivedSymbolSequence .* w_mmse;
DetectionBitSequence_MMSE = qamdemod(DetectionSymbolSequence_MMSE.', M, '
    OutputType', 'bit', 'UnitAveragePower', 1)';
```

1.4 MLD(Maximum Likelihood Detection)

```
alphabet = qammod([0:M-1], M, 'UnitAveragePower', true);
arg = (ones(length(alphabet),1) * ReceivedSymbolSequence) - (alphabet.' *
    H);
```

```
arg = abs(arg).^2;
[val,idx] = min(arg);
DetectionBitSequence_MLD = reshape(de2bi(idx-1, log2(M), 'left-msb')', 1,
    []);
```

2 결과 및 분석

2.1 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;
```

위의 *Code 1*과 *Code 2*는 동일한 결과를 이룬다.

$M = 2^2n$ $(n = 1, 2, 3, ...)$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. M개의 *alphabet*이 다음과 같다고하자.

$$alphabet = \pm(2n-1) \pm j \cdot (2n-1) \qquad n \in 1, 2, ..., \frac{\sqrt{M}}{2} \tag{6}$$

그렇다면 신호의 평균전력은 다음과 같이 일반화 가능하다.

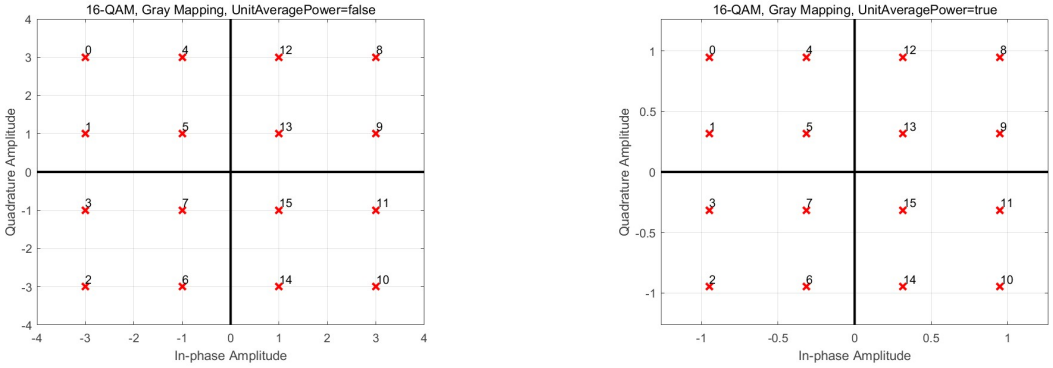
$$\begin{aligned}
E_s = E[|s|^2] &= \frac{1}{M}\sum_{n=1}^{M}|s_n|^2 \\
&= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2 + (2m-1)^2] \\
&= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2] + \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2m-1)]^2 \\
&= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} \sum_{m=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2] \cdot 2 \\
&= \frac{1}{M} \cdot 4 \sum_{n=1}^{\frac{\sqrt{M}}{2}} [(2n-1)^2 \cdot \sqrt{M}] \\
&= \frac{4}{\sqrt{M}} \sum_{n=1}^{\frac{\sqrt{M}}{2}} [4n^2 - 4n + 1] \\
&= \frac{2}{3}(M-1)
\end{aligned} \tag{7}$$

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

$$normalized\ alphabet = \Big[\pm \frac{2n-1}{\sqrt{\frac{2}{3}(M-1)}} \pm j \cdot \frac{2n-1}{\sqrt{\frac{2}{3}(M-1)}} \Big] \qquad n \in \{1, 2, ..., \sqrt{M}\} \tag{8}$$

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

참고자료
다음은 16-QAM의 Constellation이다.



(a) Non-normalized Constellation (b) Normalized Constellation

Figure 1: 16-QAM Constellation

2.2 M=2,4일 때 ZF, MMSE, MLD의 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.

$$\hat{s} = \underset{s}{\operatorname{argmin}} |z - s|^2 \tag{9}$$

$$z = w(hs + n) \tag{10}$$

$$w_{ZF} = (h)^{-1} \tag{11}$$

$$w_{MMSE} = (|h|^2 + 1/\rho)^{-1} h^* \tag{12}$$

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

ZF의 Post-processing Signal

$$\begin{aligned} z_{ZF} &= w_{ZF}(hs + n) \\ &= h^(-1)(hs + n) \\ &= s + \frac{n}{h} \end{aligned} \tag{13}$$

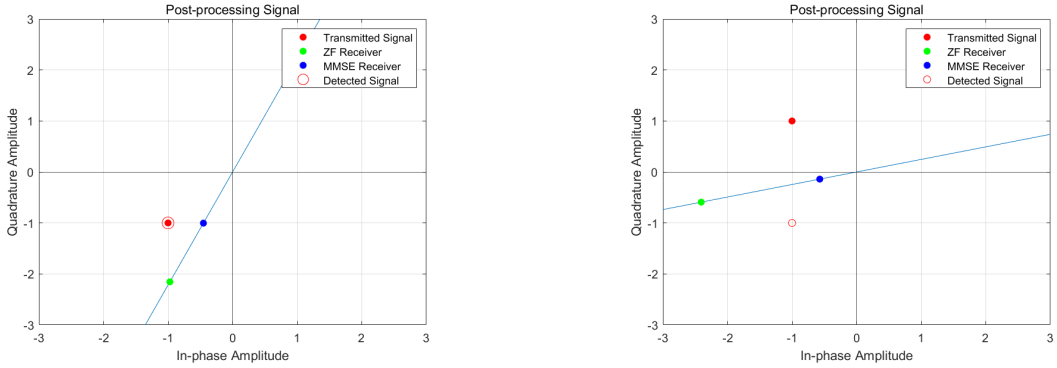
MMSE의 Post-processing Signal

$$\begin{aligned} z_{MMSE} &= w_{MMSE}(hs + n) \\ &= (|h|^2 + 1/\rho)^{-1} h^* (hs + n) \\ &= (|h|^2 + 1/\rho)^{-1} h^* \, h(s + \frac{n}{h}) \end{aligned} \tag{14}$$

위의 결과를 토대로 다음과 같이 표현이 가능하다.

$$\begin{aligned} z_{MMSE} &= (|h|^2 + 1/\rho)^{-1} h^* h \cdot z_{ZF} \\ \text{sgn}(z_{MMSE}) &= \text{sgn}((|h|^2 + 1/\rho)^{-1} h^* h \cdot z_{ZF}) \\ &= \text{sgn}(z_{ZF}) \quad (\because (|h|^2 + 1/\rho)^{-1} h^* h \geq 0) \end{aligned} \tag{15}$$

$\text{sgn}(z_{MMSE}) = \text{sgn}(z_{ZF})$라는 것은 극 좌표계로 나타냈을 때의 각도가 같다는 것을 의미한다. 이는 amplitude 보다는 phase에 영향을 받는 binary modulation이나 4-QAM의 경우, 같은 h, ρ, s의 값이 주어졌을 때 감지되는 signal은 ZF, MMSE 종류와 관계없이 동일해지는 결과를 야기한다.



(a) Correct Detection Example

(b) Signal Detection Error Example

Figure 2: 4-QAM의 ZF, MMSE Post-processing Signal Example

Figure 3는 *Matlab*을 통해 z_{MMSE}와 z_{ZF}를 직교좌표계로 바꾼 뒤, 두 θ 값을 비교한 것이다. θ값의 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 Receiver*와 *MMSE Receiver*가 하나의 직선 위에 나타난 것을 통해 이를 시각적으로도 확인할 수 있다.

```
>> [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 3: EsN0=5dB

3 과제 외적 의문점 및 질문

- SNR이 얼마나 커져야 **ZF**와 **MMSE**의 *BER*이 같아질까? 수학적 공식으로 나타낼 수 있는가?

4 Entire Code

```
1 close all
2 clear
```

```
clc

% Simulation
M = 16
Nt = 1;
NumberOfSignals = 10^2;
LengthBitSequence = Nt * NumberOfSignals*log2(M); % log2(M) bits per
    signal

NumberIteration = 10^3;

Es = 1;

EsN0_dB = -2:2:20;
EsN0 = db2pow(EsN0_dB);

EbN0 = EsN0 / log2(M);
EbN0_dB = pow2db(EbN0);

ErrorCount_ZF = zeros(1, length(EbN0_dB));
ErrorCount_MMSE = zeros(1, length(EbN0_dB));
ErrorCount_MLD = zeros(1, length(EbN0_dB));

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

for iTotal = 1 : NumberIteration
    BitSequence = randi([0 1], 1, LengthBitSequence); % Bit Generation (
        BitSequence = rand(1, LengthBitSequence) > 0.5;)
    SymbolSequence = qammod(BitSequence.', M, 'InputType', 'bit', '
        UnitAveragePower', 1).';
    %avgPower = mean(abs(SymbolSequence).^2)
    NoiseSequence = (randn(1, length(SymbolSequence)) + 1j * randn(1,
        length(SymbolSequence))) / sqrt(2); % Noise (n) Generation
    H = (randn(1, length(SymbolSequence)) + 1j * randn(1, length(
        SymbolSequence))) ./ sqrt(2); % Channel (h) Generation
    for indx_EbN0 = 1 : length(EbN0)
        ReceivedSymbolSequence = H .* SymbolSequence + NoiseSequence *
            sqrt(1 / EsN0(indx_EbN0)); % Received Signal (y = s + n)
            Generation

        % ZF Receiver
        w_zf = H.^(-1);
        DetectionSymbolSequence_ZF = ReceivedSymbolSequence .* w_zf; %
            Detection (Zero-Forcing: y / h)

        % MMSE Receiver
        w_mmse = (H.*conj(H)+1/EsN0(indx_EbN0)).^(-1) .* conj(H);
        z = ReceivedSymbolSequence .* w_mmse;
        arg = (ones(length(alphabet),1) * z) - (alphabet.' * H .* w_mmse);
        arg = arg .* conj(arg);
        [val,idx] = min(arg);
        DetectionSymbolSequence_MMSE = alphabet(idx); % TODO: could
            possibly simplify it more
        DetectionSymbolSequence_MMSE = z;
```

```
48
49          % MLD Receiver;
50          arg = (ones(length(alphabet),1) * ReceivedSymbolSequence) - (
                alphabet.' * H);
51          arg = abs(arg).^2;
52          [val,idx] = min(arg);
53          DetectionSymbolSequence_MLD = alphabet(idx);
54
55          % Symbol Sequence -> Bit Sequence
56          DetectionBitSequence_ZF = qamdemod(DetectionSymbolSequence_ZF.', M
                , 'OutputType', 'bit', 'UnitAveragePower', 1)'; % Detection
57          DetectionBitSequence_MMSE = qamdemod(DetectionSymbolSequence_MMSE
                .', M, 'OutputType', 'bit', 'UnitAveragePower', 1)'; % tmp
                value;
58          DetectionBitSequence_MLD = qamdemod(DetectionSymbolSequence_MLD.',
                 M, 'OutputType', 'bit', 'UnitAveragePower', 1)';
59
60          ErrorCount_ZF(1, indx_EbN0) = ErrorCount_ZF(1, indx_EbN0) + sum(
                DetectionBitSequence_ZF~=BitSequence);
61          ErrorCount_MMSE(1, indx_EbN0) = ErrorCount_MMSE(1, indx_EbN0) +
                sum(DetectionBitSequence_MMSE~=BitSequence);
62          ErrorCount_MLD(1, indx_EbN0) = ErrorCount_MLD(1, indx_EbN0) + sum(
                DetectionBitSequence_MLD~=BitSequence);
63      end
64  %     toc
65
66  end
67
68  BER_Simulation_ZF = ErrorCount_ZF / (LengthBitSequence * NumberIteration);
69  BER_Simulation_MMSE = ErrorCount_MMSE / (LengthBitSequence *
        NumberIteration);
70  BER_Simulation_MLD = ErrorCount_MLD / (LengthBitSequence * NumberIteration
        );
71
72  if M==2
73      BER_Theory = berfading(EbN0_dB, 'psk', 2, 1);
74  else
75      BER_Theory = berfading(EbN0_dB, 'qam', M, 1); % not sure if 'dataenc'
            needs to be specified; I don't even know what it does
76  end
77
78  % Plot
79  figure()
80  semilogy(EsN0_dB, BER_Theory, 'r--');
81  hold on
82  semilogy(EsN0_dB, BER_Simulation_ZF, 'bo');
83  semilogy(EsN0_dB, BER_Simulation_MMSE, 'bx');
84  semilogy(EsN0_dB, BER_Simulation_MLD, 'b^');
85
86
87  axis([-2 20 10^-3 0.5])
88  grid on
89  legend('Theory (Rayleigh)', 'ZF (Rayleigh)', 'MMSE (Rayleigh)', 'MLD (
        Rayleigh)');
```

```
xlabel('Es/No [dB]');
ylabel('BER');
title('BER for QAM (M='+string(M)+')');
```