

# Simulating MRC, MRT, Alamouti in a Rayleigh Fading, MIMO Environment

Project #4

Intelligent Communication Systems (ICS) Lab.  
노용재

Winter Intern Seminar (2023-1)

## Contents

<b>1</b>	<b>Implementation</b>	<b>1</b>
1.1	SIMO: Maximum Ratio Combining(MRC) . . . . .	1
1.2	Channel Unknown: Alamouti Scheme . . . . .	2
1.2.1	MISO ( $N_T=2, N_R=1$ ) . . . . .	2
1.2.2	MIMO ( $N_T=2, N_R=2$ ) . . . . .	2
1.3	Channel Known: Maximum Ratio Transmission (MRT) . . . . .	3
1.3.1	MISO ( $N_T=2, N_R=1$ ) . . . . .	3
1.3.2	MIMO ( $N_T=2, N_R=2$ ) . . . . .	3
<b>2</b>	<b>결과 및 분석</b>	<b>5</b>
2.1	Simulation Result . . . . .	5
2.1.1	BPSK . . . . .	5
2.1.2	4-QAM . . . . .	6
2.1.3	16-QAM . . . . .	7
2.2	결과 분석 . . . . .	7
<b>3</b>	<b>개선사항</b>	<b>7</b>
<b>4</b>	<b>Entire Code</b>	<b>7</b>

---

## 1 Implementation

### 1.1 SIMO: Maximum Ratio Combining(MRC)

For simplicity, let's assume  $N_T = 1, N_R = 2$ . The channel matrix  $\mathbf{H}$  would look like the following.

$$\mathbf{H} = \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix} \quad (1)$$

The received symbol will look something like this.

$$\mathbf{y} = \mathbf{H}s + \mathbf{n} \quad (2)$$

$$= \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix} s + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \quad (3)$$

By multiplying  $\mathbf{H}^H$ , we can obtain the channel gain as shown below.

$$z = \mathbf{H}^H \mathbf{y} \quad (4)$$

$$= \begin{bmatrix} h_{11}^* & h_{21}^* \end{bmatrix} \left( \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix} s + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \right) \quad (5)$$

$$= \|h\|_F^2 s + \tilde{\mathbf{n}} \quad (6)$$

```

1 NormalizationFactor = sqrt(2/3*(M-1) * Nt); % size(H,1) = Nt
2
3 y = ReceivedSymbolSequence;
4 z = H'*y;
5
6 DetectedSignal = qamdemod(z/norm(H,'fro')^2*NormalizationFactor, M);

```

## 1.2 Channel Unknown: Alamouti Scheme

### 1.2.1 MISO ( $N_T=2, N_R=1$ )

$$\begin{aligned}
z &= \mathbf{H}_{eff}^H \mathbf{y} \\
&= \begin{bmatrix} h_1^* & h_2 \\ h_2^* & -h_1 \end{bmatrix} \left( \begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{bmatrix} \mathbf{s} + \mathbf{n} \right) \\
&= \begin{bmatrix} \|h\|_F^2 & 0 \\ 0 & \|h\|_F^2 \end{bmatrix} \mathbf{s} + \tilde{\mathbf{n}}
\end{aligned} \quad (7)$$

```

1 NormalizationFactor = sqrt(2/3*(M-1) * Nt);
2 NormalizedSymbol = SymbolSequence(1:Nt, 1) / NormalizationFactor;
3
4 % Row represents antenna, Column represents time-slot
5 STBC = [NormalizedSymbol.'; -conj(NormalizedSymbol(2,1)) conj(
6     NormalizedSymbol(1,1))].';
7
8 y_alamouti = (H(1:Nr, 1:Nt) * STBC).' + NoiseSequence(1:Nt, 1) * sqrt(1 /
9     EsNO(indx_EbNO));
10
11 Augmented_H = [H; conj(H(1,2)) -conj(H(1,1))];
12
13 y = [ReceivedSymbolSequence(1,1); conj(ReceivedSymbolSequence(2,1))];
14 z = Augmented_H' * y;
15
16 FrobSquared = norm(H,'fro')^2
17 DetectedSignal = qamdemod(z/FrobSquared*NormalizationFactor, M);

```

### 1.2.2 MIMO ( $N_T=2, N_R=2$ )

$$\begin{aligned}
z &= \mathbf{H}_{eff}^H \mathbf{y} \\
&= \begin{bmatrix} h_{11}^* & h_{21}^* & h_{12} & h_{22}^* \\ h_{12}^* & h_{22}^* & -h_{11} & -h_{21}^* \end{bmatrix} \left( \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix} \mathbf{s} + \mathbf{n} \right) \\
&= \begin{bmatrix} \|h\|_F^2 & 0 \\ 0 & \|h\|_F^2 \end{bmatrix} \mathbf{s} + \tilde{\mathbf{n}}
\end{aligned} \quad (8)$$

```

1 NormalizationFactor = sqrt(2/3*(M-1) * Nt);
2
3 NormalizedSymbol = SymbolSequence(1:Nt, 1) / NormalizationFactor;
4
5 % Row prerepresents antenna, Column represents time-slot
6 STBC = [NormalizedSymbol.'; -conj(NormalizedSymbol(2,1)) conj(
    NormalizedSymbol(1,1))].';
7 Hs = reshape((H(1:Nr, 1:Nt) * STBC), [], 1);
8 y_alamouti = Hs + NoiseSequence * sqrt(1 / EsNO(indx_EbNO));
9
10 tmp_H = conj(H(:, [2 1]));
11 tmp_H(:, 2) = -tmp_H(:, 2);
12
13 Augmented_H = [H; tmp_H];
14
15 y([3 4], :) = conj(y([3 4], :));
16 z = Augmented_H' * y;
17 FrobSquared = norm(H, 'fro')^2;
18
19 DetectedSignal = qamdemod(z/FrobSquared*NormalizationFactor, M);

```

### 1.3 Channel Known: Maximum Ratio Transmission (MRT)

#### 1.3.1 MISO ( $N_T=2, N_R=1$ )

While  $\|\mathbf{w}\|_F^2 = N_T$

$$\mathbf{y} = \sqrt{\frac{E_s}{N_T}} \mathbf{h} \mathbf{w} \mathbf{s} \quad (9)$$

$$\mathbf{w} = \sqrt{N_T} \frac{\mathbf{h}^H}{\sqrt{\|\mathbf{h}\|_F^2}} \quad (10)$$

```

1 NormalizationFactor = sqrt(2/3*(M-1) * norm(H, "fro")^2);
2 w = H';
3 TransmitSymbol = w * SymbolSequence / NormalizationFactor;
4 ReceivedSymbol = H*TransmitSymbol + Noise;
5
6 FrobSquared = norm(H, 'fro')^2
7 DetectedSignal = qamdemod(ReceivedSymbol/FrobSquared*NormalizationFactor,
    M)

```

#### 1.3.2 Dominant Eigenmode Transmission MIMO ( $N_T=2, N_R=2$ )

$H$ 의 특성

SVD에 따르면 orthogonal한 행렬  $U$ 와  $V$ 에 대해 다음이 만족된다.

$$\mathbf{H} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^H \quad (11)$$

$$\mathbf{H}^H = \mathbf{V} \mathbf{\Sigma}^H \mathbf{U}^H \quad (12)$$

$$\mathbf{H} \mathbf{H}^H = \mathbf{U} \mathbf{\Sigma}^2 \mathbf{U}^H \quad (13)$$

$$\mathbf{H}^H \mathbf{H} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^H \quad (14)$$

$$(15)$$

이론

$$\mathbf{y} = \sqrt{\frac{E_s}{N_T}} \mathbf{H} \mathbf{w} s + \mathbf{n} \quad (16)$$

$$\begin{aligned} z &= \mathbf{g}^H \mathbf{y} \\ &= \sqrt{\frac{E_s}{N_T}} \mathbf{g}^H \mathbf{H} \mathbf{w} s + \mathbf{g}^H \mathbf{n} \end{aligned} \quad (17)$$

Channel gain을 최대화하기 위해  $\mathbf{g}^H = (\mathbf{H} \mathbf{w})^H$ 로 두자.

$$\begin{aligned} \mathbf{g}^H \mathbf{H} \mathbf{w} &= (\mathbf{H} \mathbf{w})^H \mathbf{H} \mathbf{w} \\ &= \mathbf{w}^H \mathbf{H}^H \mathbf{H} \mathbf{w} \\ &= \mathbf{w}^H \mathbf{V} \Sigma^2 \mathbf{V}^H \mathbf{w} \quad (\because \text{equation}(3)) \end{aligned} \quad (18)$$

2x2 MIMO에서  $\mathbf{w} = [w_1 \ w_2]^T$ 이다. 식(7)을 전개하면  $\sigma_1^2 |\mathbf{v}_1^H \mathbf{w}|^2 + \sigma_2^2 |\mathbf{v}_2^H \mathbf{w}|^2$ 이므로  $\mathbf{w}$ 의 전력이 고정돼 있을 때,  $\mathbf{v}_1 = \mathbf{w}$ 로 두는 게 최선이다.

$$\mathbf{y} = \sqrt{\frac{E_s}{N_T}} \mathbf{H} \mathbf{w} s + \mathbf{n} \quad (19)$$

$$\mathbf{z} = \mathbf{g}^H \sqrt{\frac{E_s}{N_T}} \mathbf{H} \mathbf{w} s + \mathbf{n} \quad (20)$$

$$(21)$$

If  $\mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^H$ , the best solution would be,  $\mathbf{U} = \mathbf{g}$ ,  $\mathbf{V} = \mathbf{w} / \sqrt{N_T}$  to maximize channel gain.

```

1 % S = svd(A) returns the singular values of matrix A in descending order.
2 [U,S,V] = svd(H);
3 w = V(:,1);
4 g = U(:,1);
5
6 NormalizationFactor = sqrt(2/3*(M-1));
7 TransmitSymbol = w * SymbolSequence / NormalizationFactor;
8 ReceivedSymbol = H * TransmitSymbol + Noise;
9
10 PostProcessing = (g' * ReceivedSymbol) / S(1,1);
11
12 DetectedSignal = qamdemod(PostProcessing*NormalizationFactor, M);

```

## 2 결과 및 분석

### 2.1 Simulation Result

#### 2.1.1 BPSK

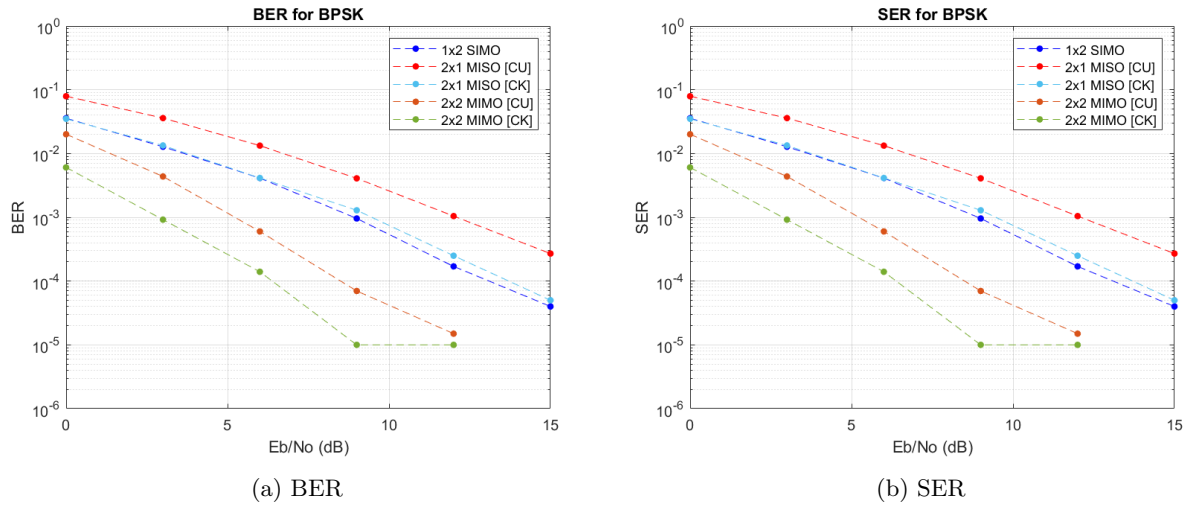
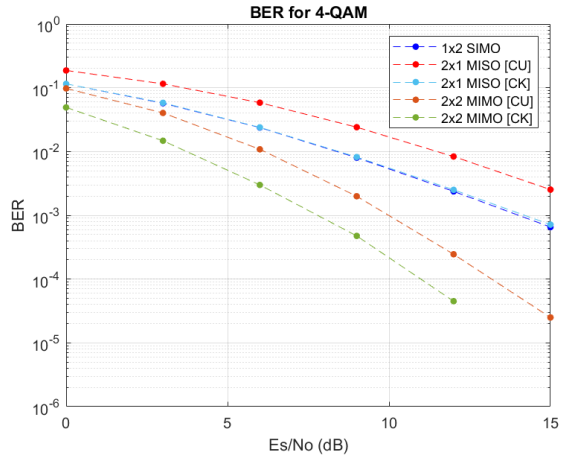
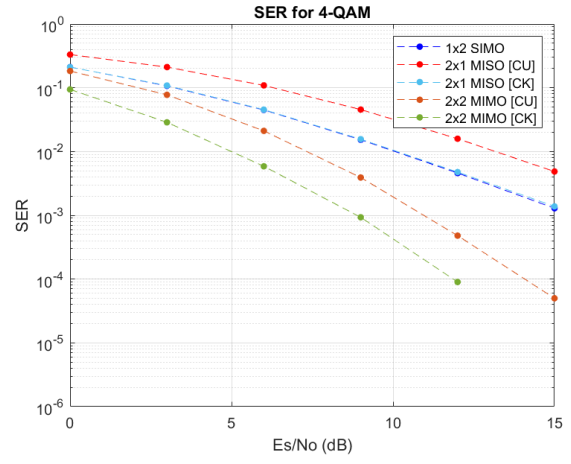


Figure 1

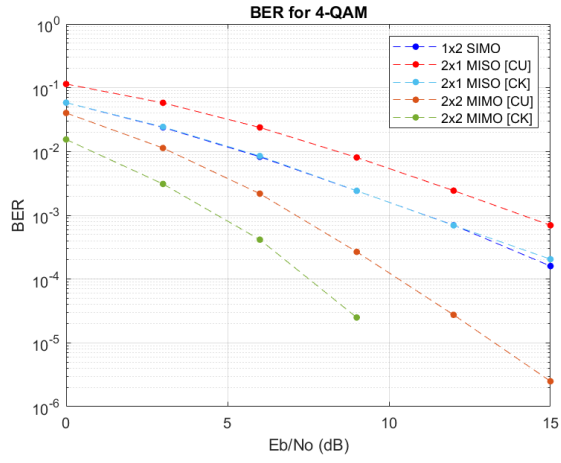
### 2.1.2 4-QAM



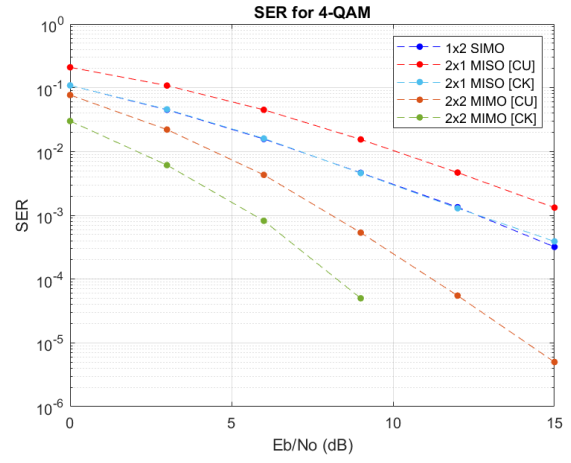
(a) BER



(b) SER



(c) BER



(d) SER

Figure 2

### 2.1.3 16-QAM

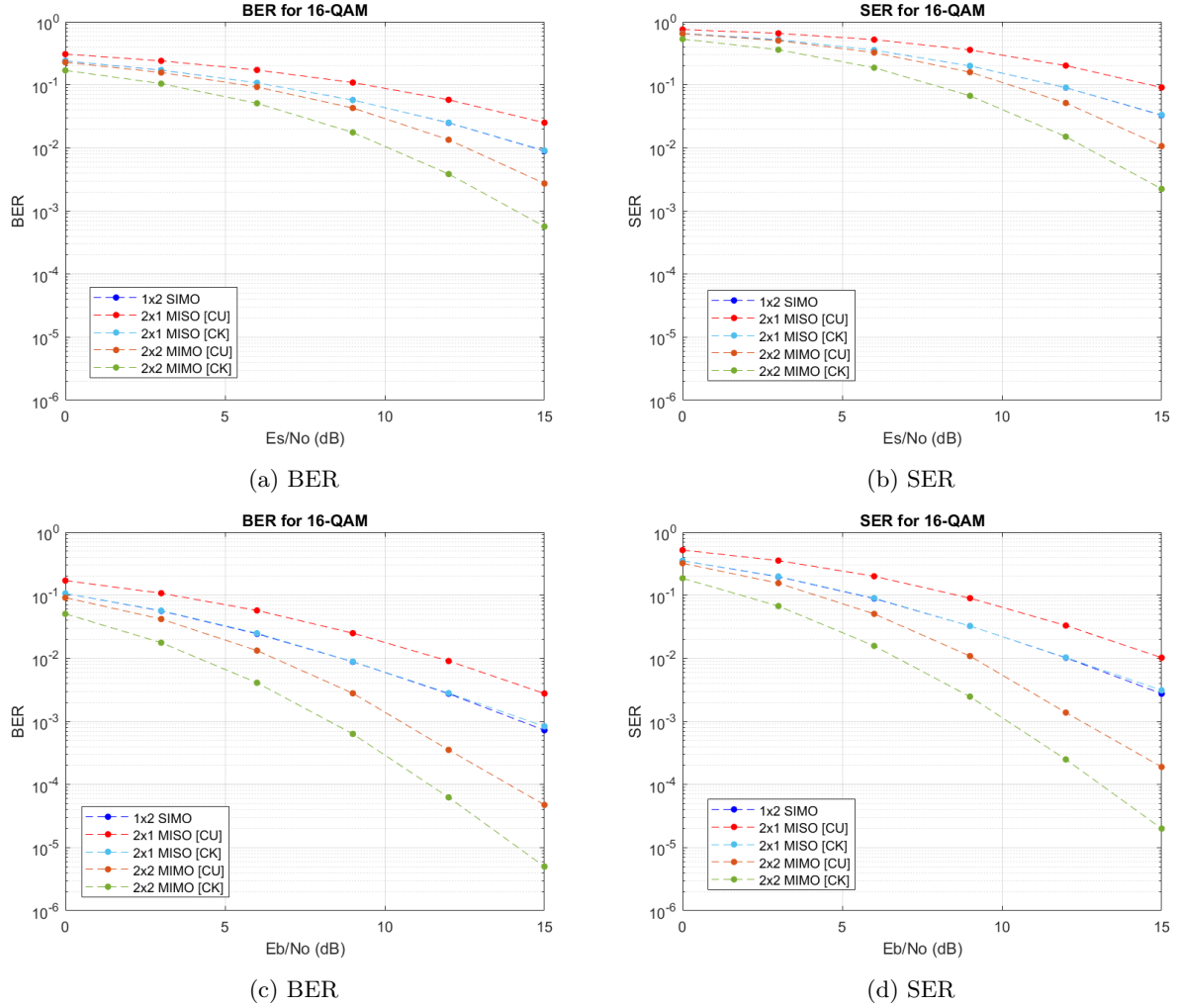


Figure 3

## 2.2 결과 분석

## 3 개선사항

- 구현은 했으나, Eigenmode decomposition 내용을 정확하게 이해하지 못해 구체적인 분석은 하지 못했다.

## 4 Entire Code <sup>1</sup>

<sup>1</sup>Uploaded on [https://github.com/lightwick/ICS\\_project/tree/main/SpatialDiversity](https://github.com/lightwick/ICS_project/tree/main/SpatialDiversity)

## main.m

```
1 close all
2 clear all
3 dbstop if error
4 % dbstop if warning
5
6 addpath(' ../tools/')
7
8 % Environment Variable
9 M = 4
10
11 NumberIteration = 10^5;
12 SimulationNum = 5;
13
14 % Simulation
15 EbNO_dB = 0:3:15;
16 EbNO = db2pow(EbNO_dB);
17
18 EsNO = EbNO * log2(M);
19 EsNO_dB = pow2db(EsNO);
20
21 % EsNO_dB = 0:3:15;
22 % EsNO = db2pow(EsNO_dB);
23 %
24 % EbNO = EsNO / log2(M);
25 % EbNO_dB = pow2db(EbNO);
26
27 % BitErrorCount_ZF = zeros(1, length(EsNO_dB));
28 % SignalErrorCount_ZF = zeros(1, length(EsNO_dB));
29 % BitErrorCount_MLD = zeros(1, length(EsNO_dB));
30 % SignalErrorCount_MLD = zeros(1, length(EsNO_dB));
31 % 33edd
32 % BitErrorCount_MMSE = zeros(1, length(EsNO_dB));
33 % SignalErrorCount_MMSE = zeros(1, length(EsNO_dB));
34
35 NormalizationFactor = sqrt(2/3*(M-1));
36
37 BitErrorCount = zeros(SimulationNum, length(EsNO_dB));
38 SignalErrorCount = zeros(SimulationNum, length(EsNO_dB));
39
40 BEC_tmp = zeros(size(BitErrorCount));
41 SEC_tmp = zeros(size(SignalErrorCount));
42
43 FivePercent = ceil(NumberIteration/20);
44
45 for iTotat = 1 : NumberIteration
46     if mod(iTotat-100, FivePercent)==0
47         tic
48     end
49     % Bit Generation
50     SignalSequence = randi([0 M-1], 2, 1);
51     SignalBinary = de2bi(SignalSequence, log2(M), 'left-msb');
52     SymbolSequence = qammod(SignalSequence, M);
53     % SymbolSequence = qammod(SignalSequence, M) / NormalizationFactor;
54
55     NoiseSequence = (randn(4, 1) + 1j * randn(4, 1)) / sqrt(2); % Noise (n) Generation
56     % NoiseSequence = zeros(4,1);
57     H = (randn(2, 2) + 1j * randn(2, 2)) ./ sqrt(2); % Receiver x Transmitter
58
59     for indx_EbNO = 1 : length(EsNO)
60         %% SIMO (MRC)
61         Nt =1;
```



```

62 Nr = 2;
63 NormalizedSymbol = SymbolSequence(1:Nt, 1) / (NormalizationFactor * sqrt(Nt));
64 Noise = NoiseSequence(1:Nr, 1) * sqrt(1 / EsNO(indx_EbNO));
65
66 y = H(1:Nr, 1:Nt) * NormalizedSymbol + Noise;
67 [BEC_tmp(1, indx_EbNO), SEC_tmp(1, indx_EbNO)] = simo_mrc(y, SignalSequence(1:Nt
68     , 1), SignalBinary(1:Nt, :), M, H(1:Nr, 1:Nt));
69
70 %% MISO (Alamouti)
71 Nt = 2;
72 Nr = 1;
73 NormalizedSymbol = SymbolSequence(1:Nt, 1) / (NormalizationFactor * sqrt(Nt));
74 % Row represents antenna, Column represents time-slot
75 STBC = [NormalizedSymbol.'; -conj(NormalizedSymbol(2,1)) conj(NormalizedSymbol
76     (1,1))].';
77 y_alamouti = (H(1:Nr, 1:Nt) * STBC).' + NoiseSequence(1:Nt, 1) * sqrt(1 / EsNO(
78     indx_EbNO));
79
80 [BEC_tmp(2, indx_EbNO), SEC_tmp(2, indx_EbNO)] = miso_alamouti(y_alamouti,
81     SignalSequence(1:Nt, 1), SignalBinary(1:Nt, :), M, H(1:Nr, 1:Nt));
82
83 %% MISO (MRT)
84 Nt = 2;
85 Nr = 1;
86 H_new = H(1:Nr, 1:Nt);
87 [BEC_tmp(3, indx_EbNO), SEC_tmp(3, indx_EbNO)] = miso_mrt(SymbolSequence(1),
88     SignalSequence(1), Noise(1:Nr), SignalBinary(1, :), M, H_new);
89
90 %% MIMO (Alamouti)
91 Nt = 2;
92 Nr = 2;
93 NormalizedSymbol = SymbolSequence(1:Nt, 1) / (NormalizationFactor * sqrt(Nt));
94 % Row represents antenna, Column represents time-slot
95 STBC = [NormalizedSymbol.'; -conj(NormalizedSymbol(2,1)) conj(NormalizedSymbol
96     (1,1))].';
97 Hs = reshape((H(1:Nr, 1:Nt) * STBC), [], 1);
98 y_alamouti = Hs + NoiseSequence * sqrt(1 / EsNO(indx_EbNO));
99 [BEC_tmp(4, indx_EbNO), SEC_tmp(4, indx_EbNO)] = mimo_alamouti(y_alamouti,
100     SignalSequence(1:Nt, 1), SignalBinary(1:Nt, :), M, H(1:Nr, 1:Nt));
101
102 %% MIMO (MRT)
103 Nt = 2;
104 Nr = 2;
105 H_new = H(1:Nr, 1:Nt);
106 Noise = NoiseSequence(1:Nr, 1) * sqrt(1 / EsNO(indx_EbNO));
107 [BEC_tmp(5, indx_EbNO), SEC_tmp(5, indx_EbNO)] = mimo_mrt(SymbolSequence(1),
108     SignalSequence(1), Noise(1:Nr), SignalBinary(1, :), M, H_new);
109
110 end
111
112 BitErrorCount = BitErrorCount + BEC_tmp;
113 SignalErrorCount = SignalErrorCount + SEC_tmp;
114
115 if mod(iTotal-100, FivePercent)==0
116     ElapsedTime = toc;
117     EstimatedTime = (NumberIteration-iTotal)*ElapsedTime;
118     disp(sprintf("%d%%, estimated wait time %d minutes %d seconds", round(iTotal/
119         NumberIteration*100), floor(EstimatedTime/60), floor(mod(EstimatedTime, 60)))
120         )
121
122 end
123
124 end
125
126 BER = BitErrorCount / (NumberIteration*log2(M));

```

```

114 SER = SignalErrorCount / (NumberIteration);
115
116 BER(2,:) = BER(2,:) / 2;
117 SER(2,:) = SER(2,:) / 2
118
119 BER(4,:) = BER(4,:) / 2;
120 SER(4,:) = SER(4,:) / 2;
121
122 % Plot
123 BER_Title = sprintf("BER for %d-QAM", M);
124 SER_Title = sprintf("SER for %d-QAM", M);
125 x_axis = "Eb/No (dB)";
126
127 legend_order = ["1x2 SIMO", "2x1 MISO [CU]", "2x1 MISO [CK]", "2x2 MIMO [CU]", "2x2 MIMO
    [CK]"];
128 myplot(EbNo_dB, BER, BER_Title, x_axis, "BER", legend_order);
129 ylim([10^(-6) 1])
130 myplot(EbNo_dB, SER, SER_Title, x_axis, "SER", legend_order);
131 ylim([10^(-6) 1])

```

#### simo\_mrc.m

```

1 function [BitErrorCount, SignalErrorCount] = simo_mrc(ReceivedSymbolSequence,
    SignalSequence, SignalBinary, M, H)
2     Nt = size(H,2);
3     Nr = size(H,1);
4     assert(Nt==1, 'Nt is not 1')
5     NormalizationFactor = sqrt(2/3*(M-1) * Nt);
6
7     y = ReceivedSymbolSequence;
8     z = H'*y;
9
10    DetectedSignal = qamdemod(z/norm(H,'fro')^2*NormalizationFactor, M);
11    DetectedBinary = de2bi(DetectedSignal, log2(M), 'left-msb');
12
13    SignalErrorCount = sum(DetectedSignal~=SignalSequence, 'all');
14    BitErrorCount = sum(SignalBinary~=DetectedBinary, 'all');
15 end

```

#### miso\_alamouti.m

```

1 function [BitErrorCount, SignalErrorCount] = miso_alamouti(ReceivedSymbolSequence,
    SignalSequence, SignalBinary, M, H)
2     Nt = size(H,2);
3     Nr = size(H,1);
4     assert(Nt==2 && Nr==1, "Need H of size 1x2")
5
6     Augmented_H = [H; conj(H(1,2)) -conj(H(1,1))];
7
8     NormalizationFactor = sqrt(2/3*(M-1) * Nt);
9
10    y = [ReceivedSymbolSequence(1,1); conj(ReceivedSymbolSequence(2,1))];
11    z = Augmented_H' * y;
12
13    FrobSquared = H*H'; % equivalent to norm(H,'fro')^2
14    DetectedSignal = qamdemod(z/FrobSquared*NormalizationFactor, M);
15    DetectedBinary = de2bi(DetectedSignal, log2(M), 'left-msb');
16
17    SignalErrorCount = sum(DetectedSignal~=SignalSequence, 'all');
18    BitErrorCount = sum(SignalBinary~=DetectedBinary, 'all');
19 end

```

#### miso\_mrt.m

```

1 function [BitErrorCount, SignalErrorCount] = miso_mrt(SymbolSequence, SignalSequence,
2 Noise, SignalBinary, M, H)
3     Nt = size(H,2);
4     Nr = size(H,1);
5     assert(Nt==2 && Nr==1, "Need H of size 1x2")
6
7     NormalizationFactor = sqrt(2/3*(M-1) * norm(H, "fro")^2);
8     w = H';
9     TransmitSymbol = w * SymbolSequence / NormalizationFactor;
10    ReceivedSymbol = H*TransmitSymbol + Noise;
11
12    FrobSquared = H*H'; % equivalent to norm(H,'fro')^2
13    DetectedSignal = qamdemod(ReceivedSymbol/FrobSquared*NormalizationFactor, M);
14    DetectedBinary = de2bi(DetectedSignal, log2(M), 'left-msb');
15
16    SignalErrorCount = sum(DetectedSignal~=SignalSequence, 'all');
17    BitErrorCount = sum(SignalBinary~=DetectedBinary, 'all');
18 end

```

#### mimo\_alamouti.m

```

1 function [BitErrorCount, SignalErrorCount] = mimo_alamouti(y, SignalSequence,
2 SignalBinary, M, H)
3     Nt = size(H,2);
4     Nr = size(H,1);
5     assert(Nt==2 && Nr==2, 'H is not size 2x2')
6     assert(length(SignalSequence), "Signal Sequence is not 2")
7
8     tmp_H = conj(H(:,[2 1]));
9     tmp_H(:,2) = -tmp_H(:,2);
10
11    Augmented_H = [H; tmp_H];
12    NormalizationFactor = sqrt(2/3*(M-1) * Nt);
13
14    y([3 4], :) = conj(y([3 4], :));
15    z = Augmented_H' * y;
16    FrobSquared = norm(H, 'fro')^2;
17
18    DetectedSignal = qamdemod(z/FrobSquared*NormalizationFactor, M);
19    DetectedBinary = de2bi(DetectedSignal, log2(M), 'left-msb');
20
21    SignalErrorCount = sum(DetectedSignal~=SignalSequence, 'all');
22    BitErrorCount = sum(SignalBinary~=DetectedBinary, 'all');
23 end

```

#### mimo\_mrt.m

```

1 function [BitErrorCount, SignalErrorCount] = mimo_mrt(SymbolSequence, SignalSequence,
2 Noise, SignalBinary, M, H)
3     Nt = size(H,2);
4     Nr = size(H,1);
5
6     % S = svd(A) returns the singular values of matrix A in descending order.
7     [U,S,V] = svd(H);
8     w = V(:,1);
9     g = U(:,1);
10
11    NormalizationFactor = sqrt(2/3*(M-1));
12    TransmitSymbol = w * SymbolSequence / NormalizationFactor;
13    ReceivedSymbol = H * TransmitSymbol + Noise;
14
15    Processing = g' * ReceivedSymbol;
16
17    DetectedSignal = qamdemod(Processing/S(1,1)*NormalizationFactor, M);

```

```
17     DetectedBinary = de2bi(DetectedSignal, log2(M), 'left-msb');
18
19     SignalErrorCount = sum(DetectedSignal~=SignalSequence, 'all');
20     BitErrorCount = sum(SignalBinary~=DetectedBinary, 'all');
21 end
```