Experiment CL1 : MIMO Communications

Demostrator: Yuan Wei Liu

Student: Haoxiang Huang   CID: 02470313

EX1

1. Set up basic Parameters

```
clear all;clc;close all;

% Basic Parameters
SNR_dB = 0:1:30; % SNR in dB
SNR_linear= power(10,SNR_dB/ 10);% SNR in linear scale
N = 5:1:10; % Number of input and output antennas at the MIMO system
num_simulations = 10000; % Number of Monte Carlo simulations
C_ergodic = zeros(length(N), length(SNR_dB));% Predefine Shannon capacity matrix
```

use Matlab to set the SNR values ranging from 0 dB to 30dB and number of antennas from 5 to 10. I also define the number of Monte Carlo simulation times and C_ergodic matrix to store result

2. Monte carlo Simulation

For MIMO system, $n_R = n_T = 2$, the Rayleigh fading channel martix can be presents as:

$$H_{ij} = \frac{X_{ij} + jY_{ij}}{2}.$$

where X_{ij}, Y_{ij} are i.i.d Gaussian r.v. with zero mean and unit variance AND their size is $n_r \times n_t$

Due to H are statistically independent over the time, for each ergodic channels, the Shannon capacity is equal to the statistical average

$$C_{ergodic} = E[C_H]$$

It is hard to derive close-form expression of MIMO capacity, so I use two nested loop to iterate over each SNR value and each number of antennas given by requirement. For each SNR and antennas, I will run 10,000 times Monte Carlo simulation to obtain the statistical average capacity

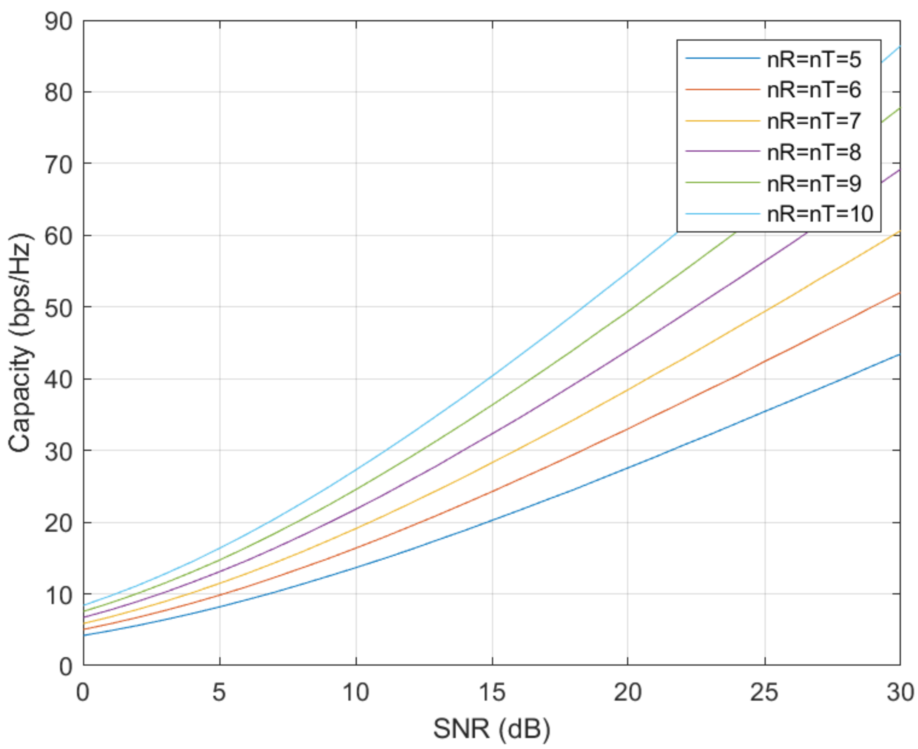
code shown below:

```
% for each defined SNR
for SNR_index = 1:length(SNR_dB)
    SNR = SNR_linear(SNR_index);

    % for each number of antennas at the MIMO system
    for N_index = 1:length(N)
        % define nR (number of receive antennas) = nT (number of transmit
        % antennas) in MIMO system
        nT = N(N_index);
        nR = nT;
        sum_capacity = 0;% Predefine Sum the capacity over all Monte Carlo simulations
        % Monte Carlo simulation
        for sim_loop = 1:num_simulations
            % Generate random Rayleigh fading channel matrix
            H = sqrt(1/2)*(randn(nR, nT) + 1i * randn(nR, nT));
            % Calculate instantaneous channel capacity in this simulation
            C_H = log2(det(eye(nR) + SNR/nT * (H' * H)));
            % Sum the capacity over all simulations
            sum_capacity = sum_capacity + C_H;
        end

        % Statistical average the capacity over all simulations
        C_ergodic(N_index, SNR_index) = sum_capacity / num_simulations;
    end
end
```

3. Plot and results analysis



We can clearly found that as the increasement of antenna number nr=nt, the capacity increase in different SNR.

EX2

1. Parameters setting up

Firstly, I initialize basic parameters like the number of antennas $nR = nT = 2$, measured SNR range, modulation order $M = 4$ [For QPSK], BER matrices etc.

$\boxed{\text{AND}}$ the simulation data size has been defined here as well

$$\begin{cases} \text{num\_bits} = 1e6 & \text{\% number of sended bits.} \\ \text{H\_rounds} = 3000 & \text{\% Simulation times for each SNR.} \end{cases}$$

Noted:

Here, I choose bigger number of random matrices H than recommended 300 in remark2 to obtain more precise and more smooth curve

Lastly, I generate random sended symbols sequence based on:

$$n_{symbol} = n_{bits} * \log_2 M$$

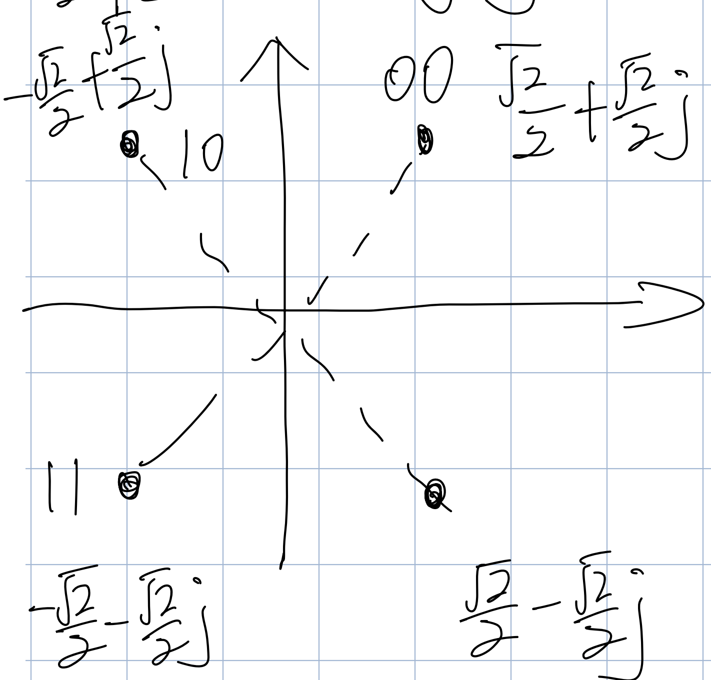
Code shown below

```
clear all;clc;close all;

% Basic Parameters
nT = 2; % Number of transmit antennas
nR = nT; % Number of receive antennas
SNR_dB_range= 0:5:35; % SNR in dB
step = 5;% SNR Iteration Step
M = 4; % QPSK modulation, so M=4
num_bits = 1e6;% Number of sended bits
H_rounds = 3000; % Number of random matrices H for each SNR
bit_symbol = log2(M); % bit per symbol
num_symbols = num_bits/ bit_symbol;% Number of sended symbols
AVG_BER_ZF = zeros(1,length(SNR_dB_range));%ZF detection Bit Error Rate
AVG_BER_ML = zeros(1,length(SNR_dB_range));%ML detection Bit Error Rate
AVG_BER_MMSE = zeros(1,length(SNR_dB_range));%MMSE detection Bit Error Rate
Tx_msg = randi([0, M-1], 1, num_symbols); % Generate random sended data
```

2. Simulation

QPSK with gray :



Due to 4PAM=QPSK.

I use qammod fuction to obtain gray mapping

```
function Rx= QPSK(M,num_symbols,SNR_dB,Tx_msg, H)
    % Perform QPSK modulation using gray code
    Tx= qammod(Tx_msg, M, 'gray','UnitAveragePower', true);

    % Reshape symbols into 2xN matrix for 2x2 MIMO
    Tx = reshape(Tx, 2, []);

    % Noise Matrix
    noise = sqrt(1/2) * (randn(2, num_symbols/2) + 1i*randn(2, num_symbols/2));

    % Receive Matrix (Y = HX + N)
    %Rx = H* Tx + noise;


    SNR_linear= power(10,SNR_dB/10);
    Rx =  sqrt(SNR_linear/ 2) *H * Tx + noise;
```

$$y = \sqrt{\frac{P}{n_T}}\, Hx + n$$

ML Detection:

$$\hat{s} = \arg\min_{s \in C} \|y - \sqrt{\tfrac{P}{n_T}} Hx\|^2 \Rightarrow$$ compare standard point to find nearest one

ZF Detection:

$$\hat{s} = Q\{(\sqrt{\tfrac{P}{n_T}} H)^{-1} y\}$$

MMSE Detection:

$$\hat{s} = Q\{(\tfrac{P}{n_T} H^H H + I)^{-1} \sqrt{\tfrac{P}{n_T}} H^H y\}$$

Here is my program structure, I define some functions:

- Parems&Images
- CL experiment.pdf
- Exercise1.m
- Exercise2.m
- Exercise3.m
- ML_Detection.m → ML detection
- ML_Detection_Alam.m
- MMSE_Detection.m → MMSE detection
- QPSK.m → QPSK modulation with gray mapping
- QPSK_Alam.m
- Quantization.m
- ZF_Detection.m → ZF detection

Here is main simulation code:

```
parfor SNR_loop= 1: 8% Multithreaded parallel computing
    SUM_BER_ML = 0;
    SUM_BER_ZF = 0;
    SUM_BER_MMSE = 0;
    SNR_dB = (SNR_loop-1)*5
    for H_loop = 1:H_rounds
        % Generate random Rayleigh fading channel matrix
        H = sqrt(1/2)*(randn(nR, nT) + 1i * randn(nR, nT));

        %QPSK modulation
        Rx = QPSK(M,num_symbols,SNR_dB,Tx_msg, H);

        % ML Detection
        ML_symbols = ML_Detection(Rx,M,H,num_symbols,nT,SNR_dB);

        % ZF Detection
        ZF_symbols = ZF_Detection(Rx,SNR_dB,H,nT);

        % MMSE Detection
        MMSE_symbols = MMSE_Detection(Rx,SNR_dB,H,nT);

        % Demodulate using QAM demodulation
        ML_Rx_msg = qamdemod(reshape(ML_symbols, 1, []), M, 'gray', 'UnitAveragePower', true);
        ZF_Rx_msg = qamdemod(reshape(ZF_symbols, 1, []), M, 'gray', 'UnitAveragePower', true);
        MMSE_Rx_msg = qamdemod(reshape(MMSE_symbols, 1, []), M, 'gray', 'UnitAveragePower', true);

        % Calculate BER
        BER_ML= sum(Tx_msg ~= ML_Rx_msg) / num_symbols;
        SUM_BER_ML = SUM_BER_ML + BER_ML;

        BER_ZF= sum(Tx_msg ~= ZF_Rx_msg) / num_symbols;
        SUM_BER_ZF = SUM_BER_ZF + BER_ZF;

        BER_MMSE= sum(Tx_msg ~= MMSE_Rx_msg) / num_symbols;
        SUM_BER_MMSE = SUM_BER_MMSE + BER_MMSE;
    end
    % Statistical average the BER
    AVG_BER_ML(SNR_loop) = SUM_BER_ML/H_rounds;
    AVG_BER_ZF(SNR_loop) = SUM_BER_ZF/H_rounds;
    AVG_BER_MMSE(SNR_loop) = SUM_BER_MMSE/H_rounds;
end
```

use "parfor" than "for" to open multi thread Pool to speed up Computation.

Symbol, Symbol modulation with gray mapping

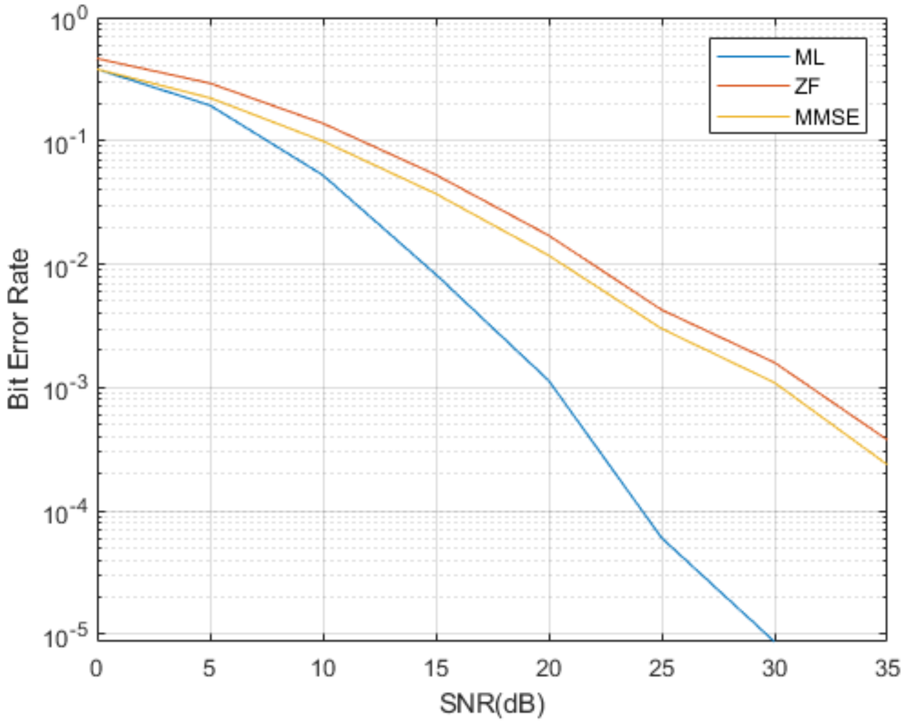
Collect each loop results.

3. result analysis

In order to get more Smooth and Precise Curve, I run 3,000 H and 10^6 bits

I find Performance:

$$ML > MMSE > ZF$$



EX3

Alamouti coding for 2 antennas

$$X = \begin{bmatrix} x_1 & -x_2^* \\ x_2 & x_1^* \end{bmatrix}$$

modify QPSK in EX2

```
function Rx= QPSK_Alam(M,num_symbols,SNR_dB,Tx_msg, H)
    % Perform QPSK modulation using gray code
    Tx= qammod(Tx_msg, M, 'gray','UnitAveragePower', true);

    % Alamouti encoding
    Tx_Alam = reshape(Tx, 2, []);
    Tx_Alam = [Tx_Alam; [-conj(Tx_Alam(2,:)); conj(Tx_Alam(1,:))]];

    % Convert the encoded symbols back to a single column for transmission
    Tx_Alam = reshape(Tx_Alam, [], 1);

    % Noise Matrix
    noise = sqrt(1/2) * ((randn(2, num_symbols) + 1i*randn(2, num_symbols)));
    SNR_linear= power(10,SNR_dB/10);
    % Receive Matrix (Y = HX + N)
    Rx = sqrt(SNR_linear/ 2) * H * reshape(Tx_Alam, 2, []) + noise;
```

keep similar main simulation code structure

Also, run 3000 H and 10^6 bits:

I find performance:

ML with Alamouti coding better than

ML without Alamouti coding.