Contents


```
%Gian Angelo Tria
%MIMO/OFDM Final Project
%Wireless Communications
%Prof. Hoerning

clc;
clear;
clear all;
```

Part 1 MIMO

```
n = 17;
M = 4;
SNR = 20;
numbits = 2^n; %Number of bits being sent
fm = [5,50,500];
for j = 1:3
for k = 1:100
%Rayleigh
for i =1:4
    H(i,:) = rayleighchan(fm(j),2);
end
H11 = H(1);
H12 = H(2);
H21 = H(3);
H22 = H(4);
H = [H11 H21; H12 H22];


bits_1 = randi([0,1], 1, numbits*M);
msg_1 = reshape(bits_1,[M,numbits]);
dec_1 = (bi2de(msg_1.','left-msb'))';
y_1 = qammod(dec_1,2^M,'UnitAveragePower',true);
bits_2 = randi([0,1], 1, numbits*M);
msg_2 = reshape(bits_2,[M,numbits]);
dec_2 = (bi2de(msg_2.','left-msb'))';
y_2 = qammod(dec_2,2^M,'UnitAveragePower',true);
data_in = [bits_1,bits_2];
%
n1 = 10^(-SNR/20)*(1/sqrt(2)*(randn(1,length(y_1)) + 1i*randn(1,length(y_1))));
n2 = 10^(-SNR/20)*(1/sqrt(2)*(randn(1,length(y_2)) + 1i*randn(1,length(y_2))));
no =[n1; n2];
y = H*[y_1;y_2]+no;

% Zero Forcing
```

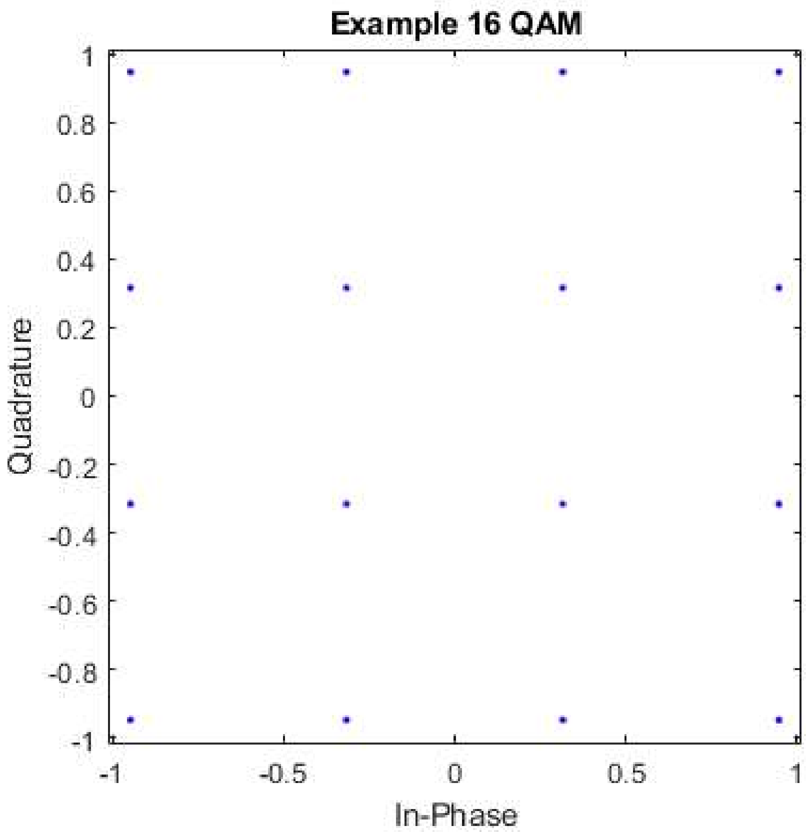
```
x_hat_1 = inv(H)*y;
rx1 = qamdemod(x_hat_1(1,:),2^M,'UnitAveragePower',true);
rx2 = qamdemod(x_hat_1(2,:),2^M,'UnitAveragePower',true);
rx_z = [rx1 rx2];
data_z_1 = de2bi(rx_z.','left-msb')';
rxMSG = reshape(data_z_1,1,[]);
[~,ber_z(k,j)] = biterr(data_in,rxMSG);

% Precoding

[U,S,V] = svd(H);
x_p = V * [y_1;y_2];
y_p = H*x_p+no;
y_hat_p = U'*y_p;
y_p_2 = inv(S)*y_hat_p;

rx1 = qamdemod(y_p_2(1,:),2^M,'UnitAveragePower',true);
rx2 = qamdemod(y_p_2(2,:),2^M,'UnitAveragePower',true);
rx_p = [rx1 rx2];
data_p_1 = de2bi(rx_p.','left-msb')';
rxMSG = reshape(data_p_1,1,[]);
[~,ber_p(k,j)] = biterr(data_in,rxMSG);

% MMSE
No = [var(n1) var(n2)];
H_H = transpose(conj(H));
W = inv(H_H*H + No*eye(2))*H_H;
y_mmse = W * y;

rx1 = qamdemod(y_mmse(1,:),2^M,'UnitAveragePower',true);
rx2 = qamdemod(y_mmse(2,:),2^M,'UnitAveragePower',true);
rx_m = [rx1 rx2];
data_mmse_1 = de2bi(rx_m.','left-msb')';
rxMSG = reshape(data_mmse_1,1,[]);
[~,ber_mmse(k,j)] = biterr(data_in,rxMSG);
end
end

zero_forcing = transpose(mean(ber_z));
precoding = transpose(mean(ber_p));
MMSE = transpose(mean(ber_mmse));
DopplerShift = [5;50;500];

T_1 = table(DopplerShift,zero_forcing,precoding,MMSE)
```

Part 2 OFDM

```
%Zero Forcing
cyclic_prefix = 64;
bits_1 = randi([0,1], 1, numbits*M);
msg_1 = reshape(bits_1,[M,numbits]);
dec_1 = (bi2de(msg_1.','left-msb'))';
y_1 = qammod(dec_1,2^M,'UnitAveragePower',true);
scatterplot(y_1)
title('Example 16 QAM')
%IFFT
```

```
ifourier = ifft(y_1);
%Adding Cyclic Prefix
cyclic = zeros(1,length(ifourier)+(2*cyclic_prefix));
%Rayleigh and noise
cyclic(cyclic_prefix+1:cyclic_prefix+length(ifourier)) = H11*ifourier;
noisy = awgn(cyclic,SNR,'measured');
%Removing Cyclic Prefix
ifourier_2 = noisy(cyclic_prefix+1:cyclic_prefix+length(ifourier));
%FFT
unrayleigh = (1/H11)*ifourier_2;
fourier = fft(unrayleigh);
% QAMDEMOD
rx_ofdm = qamdemod(fourier,2^M,'UnitAveragePower',true);
data_ofdm_1 = de2bi(rx_ofdm.','left-msb')';
ofdm_MSG = reshape(data_ofdm_1,1,[]);
[~,BER_ofdm_z] = biterr(bits_1,ofdm_MSG);


% Precoding
[U,S,V] = svd(H11);
x_p = V * ifourier;
cyclic(cyclic_prefix+1:cyclic_prefix+length(ifourier)) = V*ifourier;
noisy = awgn(H11*cyclic,SNR,'measured');
y_hat_p = U'*noisy;
y_p_2 = inv(S)*y_hat_p;
%Removing Cyclic Prefix
ifourier_2 = y_p_2(cyclic_prefix+1:cyclic_prefix+length(ifourier));
%FFT
fourier = fft(ifourier_2);
% QAMDEMOD
rx_ofdm_2 = qamdemod(fourier,2^M,'UnitAveragePower',true);
data_ofdm_2 = de2bi(rx_ofdm_2.','left-msb')';
ofdm_MSG_2 = reshape(data_ofdm_2,1,[]);
[~,BER_ofdm_p] = biterr(bits_1,ofdm_MSG_2);


%MMSE
cyclic(cyclic_prefix+1:cyclic_prefix+length(ifourier)) = H11*ifourier + n1;
No = (var(n1));
H_H = transpose(conj(H11));
W = (1/(H_H*H11 + No))*H_H;
y_mmse = W*cyclic;
ifourier_2 = y_mmse(cyclic_prefix+1:cyclic_prefix+length(ifourier));
%FFT
fourier = fft(ifourier_2);
% QAMDEMOD
rx_ofdm_3 = qamdemod(fourier,2^M,'UnitAveragePower',true);
data_ofdm_3 = de2bi(rx_ofdm_3.','left-msb')';
ofdm_MSG_3 = reshape(data_ofdm_3,1,[]);
[~,BER_ofdm_mmse] = biterr(bits_1,ofdm_MSG_3);


Bit_Error_Rate = [BER_ofdm_z;BER_ofdm_p;BER_ofdm_mmse];
eqtype = {'Zero Forcing';'Precoding';'MMSE'};
T_2 = table(eqtype,Bit_Error_Rate)
```

```
T_2 =

  3×2 table

        eqtype           Bit_Error_Rate
    ______________       ______________

    'Zero Forcing'                0
    'Precoding'                   0
    'MMSE'                  0.49777
```



OFDM + MIMO

```
cyclic_prefix = 64;
SNR = 60;
for j = 1:3
for k = 1:100
%Rayleigh

for i =1:4
    H(i,:) = rayleighchan(fm(j),2);
end
H11 = H(1);
H12 = H(2);
H21 = H(3);
H22 = H(4);
H = [H11 H21; H12 H22];
```

```
bits_1 = randi([0,1], 1, numbits*M);
msg_1 = reshape(bits_1,[M,numbits]);
dec_1 = (bi2de(msg_1.','left-msb'))';
y_1 = qammod(dec_1,2^M,'UnitAveragePower',true);
bits_2 = randi([0,1], 1, numbits*M);
msg_2 = reshape(bits_2,[M,numbits]);
dec_2 = (bi2de(msg_2.','left-msb'))';
y_2 = qammod(dec_2,2^M,'UnitAveragePower',true);
data_in = [bits_1,bits_2];

ifourier_1 = ifft(y_1);
cyclic_1 = zeros(1,length(ifourier_1)+(2*cyclic_prefix));
ifourier_2 = ifft(y_2);
cyclic_2 = zeros(1,length(ifourier_1)+(2*cyclic_prefix));

cyclic_1(cyclic_prefix+1:cyclic_prefix+length(ifourier_1)) = ifourier_1;
cyclic_2(cyclic_prefix+1:cyclic_prefix+length(ifourier_2)) = ifourier_2;
cyclic = [cyclic_1;cyclic_2];


n1 = 10^(-SNR/20)*(1/sqrt(2)*(randn(1,length(cyclic_1)) + 1i*randn(1,length(cyclic_1))));
n2 = 10^(-SNR/20)*(1/sqrt(2)*(randn(1,length(cyclic_2)) + 1i*randn(1,length(cyclic_2))));
no =[n1; n2];
y = H*cyclic + [n1;n2];

% Zero Forcing
x_hat_1 = inv(H)*y;
n_c_1 = x_hat_1(:,cyclic_prefix+1:cyclic_prefix+length(ifourier_1));
fourier_1 = fft(n_c_1(1,:));
fourier_2 = fft(n_c_1(2,:));

% QAMDEMOD
rx_ofdm_final_1 = qamdemod(fourier_1,2^M,'UnitAveragePower',true);
data_ofdm_final_1 = de2bi(rx_ofdm_final_1.','left-msb')';
ofdm_MSG_final_1 = reshape(data_ofdm_final_1,1,[]);
rx_ofdm_final_2 = qamdemod(fourier_2,2^M,'UnitAveragePower',true);
data_ofdm_final_2 = de2bi(rx_ofdm_final_2.','left-msb')';
ofdm_MSG_final_2 = reshape(data_ofdm_final_2,1,[]);
output = [ofdm_MSG_final_1,ofdm_MSG_final_2];
[~,BER_ofdm_final(k,j)] = biterr(data_in,output);
end
end
zero_forcing_final = transpose(mean(BER_ofdm_final));
T_3 = table(DopplerShift,zero_forcing_final)
```

Rayleigh

```
function [out] = rayleighchan(fm,numbits)
%Making gaussian distribution but negative frequencies are conjugated
%Shifted by fm to work with matlab matrix notation
gauss_0 = randn(1,numbits/2) + 1i.*randn(1,numbits/2);
conj_0 = fliplr(conj(gauss_0));
gauss_0_new = [conj_0, gauss_0];
gauss_1 = randn(1,numbits/2) + 1i.*randn(1,numbits/2);
```

```
conj_1 = fliplr(conj(gauss_1));
gauss_1_new = [conj_1, gauss_1];

%Computing doppler spectrum at baseband so fc = 0
f = linspace(-fm*(.999),fm*(.999),numbits);
dopp_spectrum = (1.5)./ ((pi*fm)*sqrt(1-(f./fm).^2));
%Multiply them together
gauss_dopp_0 = gauss_0_new.*sqrt(dopp_spectrum);
gauss_dopp_1 = gauss_1_new.*sqrt(dopp_spectrum);
%The ends have a value of infinity so I put them to the adjacent value instead
gauss_dopp_0([1 end]) = gauss_dopp_0(2);
gauss_dopp_1([1 end]) = gauss_dopp_1(2);

ifft0 = ifft(gauss_dopp_0,numbits);
ifft1 = ifft(gauss_dopp_1,numbits);
out = sqrt(ifft0.^2 + ifft1.^2);
end
```

```
T_1 =

  3×4 table
```

DopplerShift	zero_forcing	precoding	MMSE
5	0.0075076	0.0066266	0.0075475
50	0.031491	0.028326	0.033583
500	0.15184	0.11509	0.16255

Published with MATLAB® R2018b