ECE 233 - Spring 2020

Wireless Communications System Design, Modeling, and Implementation

Assignment 2

By: Swapnil Sayan Saha

UID: 605353215

Date of Submission: 23rd May 2020

Entire MATLAB Code for All Sections:

P.S. Due to the nature of character formatting in LaTeX, it's best to not copy paste the code directly from LaTeX pdf to MATLAB as it yields in formatting errors. Rather, you can download the same code as .m file directly from the following URL and use for verification:
https://drive.google.com/file/d/1Cz84U0wNIQtaXoyBz5H9aW_m3Xf9jrha/view?usp=sharing

```
clear; clc; close all;

%% Problem 1: Antenna Spacing and Beamforming

N = 16; %number of antenna
f_c = 2.4e9; %carrier frequency, 2.4 GHz (assumption)
phi = deg2rad(-30); %Steering angle (radians)
lambda = 3e8/f_c; %wavelength
d = [lambda/2; 2*lambda; lambda/8]; %antenna spacing
theta = -0.5*pi:0.01:0.5*pi;  %polar plot range
figure
for j = 1:length(d)
    f_phi = (1/sqrt(N)).*(exp(-1i.*2.*pi.*((1:N)-1).*(d(j)/lambda).*sin(phi))); ...
        %beamforming vector
    gain = zeros(1,length(theta));
    for t = 1:length(theta)
       a_theta = ...
           (1/sqrt(N)).*(exp(-1i.*2.*pi.*((1:N)-1).*(d(j)/lambda).*sin(theta(t)))); ...
           %ULA array response
       gain(t) = abs(f_phi*a_theta')^2; %array gain
    end
    polarplot(theta, 10*log10(gain), 'LineWidth', 1) %plot on polar coordinates
    hold on
end
legend('d = \lambda/2', 'd = 2\lambda', 'd = \lambda/8')
title('Plot of normalized beam pattern for various antenna spacing');
rlim([-10 0]); %set radius axis limits from -10 to 0
thetalim([-90 90]); %set theta axis limits from -pi/2 to pi/2
saveas(gcf,'P1_Q1','epsc'); %eps offers much higher resolution than jpg

%% Problem 2, Question 1: Channel Capacity of MIMO System

num_trials = 1000; %1000 MC trials
N_t = 32; %number of transmit antennae
N_r = 4:2:32; %number of receiver antennae
d_t = lambda/2; %transmit antenna spacing
d_r = lambda/2; %receiver antenna spacing
L = 4; %number of multipath components
P_t = 1; %transmit power in watts
```

```
B = 200000; %bandwidth of channel
N0 = -174; %noise spectral density
cr = zeros(num_trials,length(N_r));
cs = zeros(num_trials,length(N_r));
for j = 1:length(N_r)
    for k = 1:num_trials
        H_r = sqrt(1/2)*(randn([N_r(j),N_t]) + (1i*randn([N_r(j),N_t]))); %generate ...
            H_r (rich scattering) from complex gaussian distribution (mean 0, ...
            variance 1)
        H_s = zeros(N_r(j),N_t);
        for m = 1:L
            theta_i = unifrnd(-pi/2,pi/2); %generate AoA from uniform distribution
            phi_i = unifrnd(-pi/2,pi/2); %generate AoD from uniform distribution
            alpha_i = sqrt(max(N_t,N_r(j))/2)*(randn+(1i*randn)); %generate path ...
                gain from complex gaussian distribution (mean 0, variance: max of ...
                (Nt, Nr))
            a_rx_theta_i = ...
                ((1/sqrt(N_r(j))).*exp(-1i.*2.*pi.*((1:N_r(j))-1).*(d_r./lambda).*sin(theta_i)))'; ..
                %receiver spatial response vector
            a_tx_phi_i = ...
                ((1/sqrt(N_t)).*exp(-1i.*2.*pi.*((1:N_t)-1).*(d_r./lambda).*sin(phi_i)))'; ...
                %transmitter spatial response vector
            H_s = H_s + alpha_i.*(a_rx_theta_i*a_tx_phi_i'); %generate Hs (sparse ...
                scattering)
        end
        S_r = svd(H_r); %singular values of H_r
        S_s = svd(H_s); %singular values of H_s
        K_r = rank(H_r); %channel rank of H_r
        K_s = rank(H_s); %channel rank of H_s
        for o = 1:K_r
            cr(k,j) = cr(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(K_r.*B.*(10^-3).*db2mag(N0)^2)).*abs(S_r(o))^2)); ...
                %channel capacity for H_r
        end
        for o = 1:K_s
            cs(k,j) = cs(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(K_s.*B.*(10^-3).*db2mag(N0)^2)).*abs(S_s(o))^2)); ...
                %channel capacity for H_s
        end
    end
end
cr_avg = mean(cr,1);
cs_avg = mean(cs,1);
%plotting in db to make the change in sparse scattering a little more
%visible.
figure, semilogy(N_r,10*log10(cr_avg),'LineWidth',1.5,'Marker','*'), title('Channel ...
    capacities for rich and sparse scattering'), grid minor, xlabel('N_r'), ...
```

```
    ylabel('Channel Capacity (dB)');
hold on
semilogy(N_r,10*log10(cs_avg),'LineWidth',1.5,'Marker','o')
hold off
xlim([4 32]);
ylim([76.5 78]);
legend('Rich scattering', 'Sparse scattering')
axes('Position',[0.4 0.5 .5 .3])
semilogy(N_r,cr_avg./(10^6),'LineWidth',1.5,'Marker','*'), grid minor, ...
    xlabel('N_r'), ylabel('Channel Capacity (Mbps)');
hold on
semilogy(N_r,cs_avg./(10^6),'LineWidth',1.5,'Marker','o')
hold off
xlim([4 32]);
saveas(gcf,'P2_Q1','epsc');

%% Problem 2, Question 2: Achievable Rate
%channel matrix known
P = 2;
ar = zeros(num_trials,length(N_r)); %matrix to store achievable rates for rich ...
    scattering (channel matrix known)
as = zeros(num_trials,length(N_r)); %matrix to store achievable rates for sparse ...
    scattering (channel matrix known)
for j = 1:length(N_r)
    for k = 1:num_trials
        H_r = sqrt(1/2)*(randn([N_r(j),N_t]) + (1i*randn([N_r(j),N_t]))); %generate ...
            H_r (rich scattering) from complex gaussian distribution (mean 0, ...
            variance 1)
        H_s = zeros(N_r(j),N_t);
        for m = 1:L
            theta_i = unifrnd(-pi/2,pi/2); %generate AoA from uniform distribution
            phi_i = unifrnd(-pi/2,pi/2); %generate AoD from uniform distribution
            alpha_i = sqrt(max(N_t,N_r(j))/2)*(randn+(1i*randn)); %generate path ...
                gain from complex gaussian distribution (mean 0, variance: max of ...
                (Nt, Nr))
            a_rx_theta_i = ...
                ((1/sqrt(N_r(j))).*exp(-1i.*2.*pi.*((1:N_r(j))-1).*(d_r./lambda).*sin(theta_i)))'; ..
                %receiver spatial response vector
            a_tx_phi_i = ...
                ((1/sqrt(N_t)).*exp(-1i.*2.*pi.*((1:N_t)-1).*(d_r./lambda).*sin(phi_i)))'; ...
                %transmitter spatial response vector
            H_s = H_s + alpha_i.*(a_rx_theta_i*a_tx_phi_i'); %generate Hs (sparse ...
                scattering)
        end
        S_r = svd(H_r); %singular values of H_r
        S_s = svd(H_s); %singular values of H_s
        for o = 1:P
```

```
            ar(k,j) = ar(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(P.*B.*(10^-3).*db2mag(N0)^2)).*abs(S_r(o))^2)); ...
                %channel capacity for H_r
        end
        for o = 1:P
            as(k,j) = as(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(P.*B.*(10^-3).*db2mag(N0)^2)).*abs(S_s(o))^2)); ...
                %channel capacity for H_s
        end
    end
end
ar_avg = mean(ar,1);
as_avg = mean(as,1);


%channel matrix unknown
ar_u = zeros(num_trials,length(N_r)); %matrix to store achievable rates for rich ...
    scattering (channel matrix unknown)
as_u = zeros(num_trials,length(N_r)); %matrix to store achievable rates for rsparse ...
    scattering (channel matrix unknown)
f_tr = -pi/2+((0:N_t-1).*(pi/N_t)); %angular set for precoding vectors
for j = 1:length(N_r)
    w_tr = -pi/2+((0:N_r(j)-1).*(pi/N_r(j))); %angular set for combining vectors
    for k = 1:num_trials
        H_r_u = sqrt(1/2)*(randn([N_r(j),N_t]) + (1i*randn([N_r(j),N_t]))); ...
            %generate H_r (rich scattering) from complex gaussian distribution ...
            (mean 0, variance 1)
        H_s_u = zeros(N_r(j),N_t);
        for m = 1:L
            theta_i_u = unifrnd(-pi/2,pi/2); %generate AoA from uniform distribution
            phi_i_u = unifrnd(-pi/2,pi/2); %generate AoD from uniform distribution
            alpha_i_u = sqrt(max(N_t,N_r(j))/2)*(randn+(1i*randn)); %generate path ...
                gain from complex gaussian distribution (mean 0, variance: max of ...
                (Nt, Nr))
            a_rx_theta_i_u = ...
                ((1/sqrt(N_r(j))).*exp(-1i.*2.*pi.*((1:N_r(j))-1).*(d_r./lambda).*sin(theta_i_u)))';
                %receiver spatial response vector
            a_tx_phi_i_u = ...
                ((1/sqrt(N_t)).*exp(-1i.*2.*pi.*((1:N_t)-1).*(d_r./lambda).*sin(phi_i_u)))'; ...
                %transmitter spatial response vector
            H_s_u = H_s_u + alpha_i_u.*(a_rx_theta_i_u*a_tx_phi_i_u'); %generate Hs ...
                (sparse scattering)
        end
        yr_opt = 0; %variable to store best y (rich scattering)
        ys_opt = 0; %variable to store best y (sparse scattering)
        for q = 1:length(f_tr)
            for r = 1:length(w_tr)
```

```
                w_vec= ...
                    ((1/sqrt(N_r(j))).*exp(-1i.*2.*pi.*((1:N_r(j))-1).*(d_r./lambda).*sin(w_tr(r))))'
                    %calculate combining vector
                f_vec = ...
                    ((1/sqrt(N_t)).*exp(-1i.*2.*pi.*((1:N_t)-1).*(d_r./lambda).*sin(f_tr(q)))).'; ...
                    %calculate precoding vector
                yr_u = abs(w_vec'*H_r_u*f_vec); %calculate signal power (rich ...
                    scattering)
                ys_u = abs(w_vec'*H_s_u*f_vec); %calculate signal power (sparse ...
                    scattering)
                %select best precoder combiner pair (pair that yields
                %highest signal power:
                if(yr_u > yr_opt)
                    yr_opt = yr_u;
                end
                if(ys_u > ys_opt)
                    ys_opt = ys_u;
                end
            end
        end
        for o = 1:P
            ar_u(k,j) = ar_u(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(P.*B.*(10^-3).*db2mag(N0)^2)).*yr_opt^2)); ...
                %channel capacity for H_r
        end
        for o = 1:P
            as_u(k,j) = as_u(k,j) + ...
                B.*log2(1+(N_t.*N_r(j).*(P_t/(P.*B.*(10^-3).*db2mag(N0)^2)).*ys_opt^2)); ...
                %channel capacity for H_s
        end
    end
end
ar_u_avg = mean(ar_u,1);
as_u_avg = mean(as_u,1);

%plot
figure, semilogy(N_r,10*log10(cr_avg),'LineWidth',1.5,'Marker','*'), title('Channel ...
    capacities and achievable rates for rich and sparse scattering'), grid minor, ...
    xlabel('N_r'), ylabel('Channel Capacity / Achievable Rates (dB)');
hold on
semilogy(N_r,10*log10(cs_avg),'LineWidth',1.5,'Marker','o')
semilogy(N_r,10*log10(ar_avg),'LineWidth',1.5,'Marker','x')
semilogy(N_r,10*log10(as_avg),'LineWidth',1.5,'Marker','+')
semilogy(N_r,10*log10(ar_u_avg),'LineWidth',1.5,'Marker','s')
semilogy(N_r,10*log10(as_u_avg),'LineWidth',1.5,'Marker','d')
legend('CC (RS)', 'CC (SS)', 'AR (channel matrix known) (RS)', 'AR (channel matrix ...
    known) (SS)', 'AR (channel matrix unknown) (RS)', 'AR (channel matrix unknown) ...
```

```
    (SS)','Location','northwest')
xlim([4 32]);
hold off
axes('Position',[0.4 0.4 .5 .3])
semilogy(N_r,ar_avg./(10^6),'LineWidth',1.5,'Marker','x','Color',[0.93,0.69,0.13]), ...
    grid minor, xlabel('N_r'), ylabel('Achievable Rates (Mbps)')
hold on
semilogy(N_r,as_avg./(10^6),'LineWidth',1.5,'Marker','+','Color',[0.49,0.18,0.56])
semilogy(N_r,ar_u_avg./(10^6),'LineWidth',1.5,'Marker','s','Color',[0.47,0.67,0.19])
semilogy(N_r,as_u_avg./(10^6),'LineWidth',1.5,'Marker','d','Color',[0.30,0.75,0.93])
xlim([4 32]);
saveas(gcf,'P2_Q2','epsc');
hold off

%% Problem 3, Question 1: Impact of DAC Quantization (frequency domain)
clear;
M= [8,32]; %num. of antennae
b = [12,4]; %number of DAC bits
phi1 = deg2rad(30); %location of UE-1
phi2 = deg2rad(40); %location of UE-2
ABset = [-1/sqrt(2), 1/sqrt(2)]; %possible A, B values for unit energy 4QAM symbols
nsym = 1000; %number of 4-QAM symbols per user
s1n = zeros(1,nsym); %matrix to store 4-QAM symbols for UE-1
s2n = zeros(1,nsym); %matrix to store 4-QAM symbols for UE-2, which is set to 0
for j = 1:nsym
    A_B =  datasample(ABset, 2); %randomly sample from set
    s1n(1,j) = A_B(1) + (1i*A_B(2)); %generate unit energy 4QAM symbol for UE-1
end
%Upsample and pulseshaping:
s1n_bar = upfirdn(upsample(s1n,4), rcosdesign(0.5,8,4,'sqrt')); %same results ...
    obtained with conv
s2n_bar = upfirdn(upsample(s2n,4), rcosdesign(0.5,8,4,'sqrt'));

%channel matrix and precoding matrix for part (a) and (b):
h1ab = exp(-1i.*pi.*((1:M(1))-1).*sin(phi1));
h2ab = exp(-1i.*pi.*((1:M(1))-1).*sin(phi2));
Hab = [h1ab',h2ab'];
Pab = Hab'*(inv(Hab*Hab'));
%channel matrix and precoding matrix for part (c) and (d):
h1cd = exp(-1i.*pi.*((1:M(2))-1).*sin(phi1));
h2cd = exp(-1i.*pi.*((1:M(2))-1).*sin(phi2));
Hcd = [h1cd',h2cd'];
Pcd = Hcd'*(inv(Hcd*Hcd'));

%output of precoders:
xab = Pab*[s1n_bar',s2n_bar'];
xcd = Pcd*[s1n_bar',s2n_bar'];
```

```
%quantization levels (partitions) for parts (a) to (d):
qa = linspace(-1/M(1),1/M(1),2^b(1));
qb = linspace(-1/M(1),1/M(1),2^b(2));
qc = linspace(-1/M(2),1/M(2),2^b(1));
qd = linspace(-1/M(2),1/M(2),2^b(2));

%codebook for quantization (1 greater than partition):
ca = zeros(1,length(qa)+1);
cb = zeros(1,length(qb)+1);
cc = zeros(1,length(qc)+1);
cd = zeros(1,length(qd)+1);
ca(1) = qa(1);
cb(1) = qb(1);
cc(1) = qc(1);
cd(1) = qd(1);
ca(end) = qa(end);
cb(end) = qb(end);
cc(end) = qc(end);
cd(end) = qd(end);
for r = 1:length(qa)-1
    ca(r+1) = (qa(r)+qa(r+1))/2;
    cc(r+1) = (qc(r)+qc(r+1))/2;
end
for r = 1:length(qb)-1
    cb(r+1) = (qb(r)+qb(r+1))/2;
    cd(r+1) = (qd(r)+qd(r+1))/2;
end

%define matrices to store y_m:
ya =zeros(size(xab));
yb =zeros(size(xab));
yc =zeros(size(xcd));
yd =zeros(size(xcd));

%Apply quantization for parts (a) to (d):
for q = 1:size(xab,1)
    [~,realvec] = quantiz(real(xab(q,:)),qa,ca);
    [~,imagvec] = quantiz(imag(xab(q,:)),qa,ca);
    ya(q,:) = realvec + 1i.*imagvec;
end
for q = 1:size(xab,1)
    [~,realvec] = quantiz(real(xab(q,:)),qb,cb);
    [~,imagvec] = quantiz(imag(xab(q,:)),qb,cb);
    yb(q,:) = realvec + 1i.*imagvec;
end
for q = 1:size(xcd,1)
```

```
    [~,realvec] = quantiz(real(xcd(q,:)),qc,cc);
    [~,imagvec] = quantiz(imag(xcd(q,:)),qc,cc);
    yc(q,:) = realvec + 1i.*imagvec;
end
for q = 1:size(xcd,1)
    [~,realvec] = quantiz(real(xcd(q,:)),qd,cd);
    [~,imagvec] = quantiz(imag(xcd(q,:)),qd,cd);
    yd(q,:) = realvec + 1i.*imagvec;
end

figure
%For question 1, z(n) = y(n). Periodograms for parts (a) to (d):

[psda, nfa] = periodogram(ya(1,:));
[psdb, nfb] = periodogram(yb(1,:));
[psdc, nfc] = periodogram(yc(1,:));
[psdd, nfd] = periodogram(yd(1,:));
plot(2.*(nfa./max(nfa)),10*log10(psda),'LineWidth',1.5), title('Periodograms for ...
    DAC Quantization Impact'), grid minor, xlabel('Normalized Frequency (x \pi ...
    rad/sample)'), ylabel('Power/frequency (db/(rad/sample))');
% the 2.*(nfa./max(nfa)) is meant to normalize the frequencies just as one
% would obtain with a plot yielded directly with periodogram
hold on
plot(2.*(nfa./max(nfb)),10*log10(psdb),'LineWidth',1.5)
plot(2.*(nfa./max(nfc)),10*log10(psdc),'LineWidth',1.5)
plot(2.*(nfa./max(nfd)),10*log10(psdd),'LineWidth',1.5)
legend('M = 8, b = 12', 'M = 8, b= 4', 'M = 32, b = 12', 'M = 32, b = 4');
hold off
saveas(gcf,'P3_Q1','epsc');

%% Problem 3, Question 2: Impact of DAC Quantization (angular domain)
phi = deg2rad(-90:0.5:90);
g_phi_a = zeros(size(phi));
g_phi_b = zeros(size(phi));
g_phi_c = zeros(size(phi));
g_phi_d = zeros(size(phi));
for w = 1:length(phi)
    a_phi_ab = (exp(-1i.*pi.*((1:M(1))-1).*sin(phi(w))))';
    a_phi_cd = (exp(-1i.*pi.*((1:M(2))-1).*sin(phi(w))))';
    for x = 1:length(ya)
        g_phi_a(1,w) = g_phi_a(1,w) + abs(a_phi_ab'*ya(:,x)).^2;
        g_phi_b(1,w) = g_phi_b(1,w) + abs(a_phi_ab'*yb(:,x)).^2;
        g_phi_c(1,w) = g_phi_c(1,w) + abs(a_phi_cd'*yc(:,x)).^2;
        g_phi_d(1,w) = g_phi_d(1,w) + abs(a_phi_cd'*yd(:,x)).^2;
    end
end
g_phi_a = (1/length(ya)).*g_phi_a;
```

```
g_phi_b = (1/length(yb)).*g_phi_b;
g_phi_c = (1/length(yc)).*g_phi_c;
g_phi_d = (1/length(yd)).*g_phi_d;
figure, plot(rad2deg(phi),g_phi_a,'LineWidth',1.5), title('g(\phi) vs \phi for MIMO ...
    system (effect of DAC quantization)'), grid minor, xlabel('\phi'), ...
    ylabel('Angular response');
hold on
plot(rad2deg(phi),g_phi_b,'-.','LineWidth',1.5);
plot(rad2deg(phi),g_phi_c,'LineWidth',1.5);
plot(rad2deg(phi),g_phi_d,'-.','LineWidth',1.5);
legend('M = 8, b = 12', 'M = 8, b= 4', 'M = 32, b = 12', 'M = 32, b = 4');
hold off
axes('Position',[0.2 0.4 .3 .4])
plot(rad2deg(phi),g_phi_a,'LineWidth',1.5), grid minor, xlabel('\phi'), ...
    ylabel('Angular response');
hold on
plot(rad2deg(phi),g_phi_b,'-.','LineWidth',1.5);
plot(rad2deg(phi),g_phi_c,'LineWidth',1.5);
plot(rad2deg(phi),g_phi_d,'-.','LineWidth',1.5);
hold off
xlim([39 41])
saveas(gcf,'P3_Q2','epsc');


%% Problem 3, Question 3: Impact of PA non-linarity in the frequency domain

beta_3 = -133;

%as beta_3 for part a and c is 0, z(n) = y(n) = x(n) for part a and c.
za_3 = xab;
zb_3 = xab + beta_3.*xab.*abs(xab).^2;
zc_3 = xcd;
zd_3 = xcd + beta_3.*xcd.*abs(xcd).^2;

[psda, nfa] = periodogram(za_3(1,:));
[psdb, nfb] = periodogram(zb_3(1,:));
[psdc, nfc] = periodogram(zc_3(1,:));
[psdd, nfd] = periodogram(zd_3(1,:));

figure, plot(2.*(nfa./max(nfa)),10*log10(psda),'LineWidth',1.5), ...
    title('Periodograms for PA non-linarity impact'), grid minor, ...
    xlabel('Normalized Frequency (x \pi rad/sample)'), ylabel('Power/frequency ...
    (db/(rad/sample))');
hold on
plot(2.*(nfa./max(nfb)),10*log10(psdb),'LineWidth',1.5)
plot(2.*(nfa./max(nfc)),10*log10(psdc),'LineWidth',1.5)
plot(2.*(nfa./max(nfd)),10*log10(psdd),'LineWidth',1.5)
```

```
legend('[M,\beta]=[8,(1,0)]', '[M,\beta]=[8,(1,-133)]', '[M,\beta]=[32,(1,0)]', ...
    '[M,\beta]=[32,(1,-133)]');
hold off
saveas(gcf,'P3_Q3','epsc');

%% Problem 3, Question 4: Impact of PA non-linarity in the angular domain
g_phi4_a = zeros(size(phi));
g_phi4_b = zeros(size(phi));
g_phi4_c = zeros(size(phi));
g_phi4_d = zeros(size(phi));
for w = 1:length(phi)
    a_phi_ab = (exp(-1i.*pi.*((1:M(1))-1).*sin(phi(w))))';
    a_phi_cd = (exp(-1i.*pi.*((1:M(2))-1).*sin(phi(w))))';
    for x = 1:length(za_3)
        g_phi4_a(w) = g_phi4_a(w) + abs(a_phi_ab'*za_3(:,x)).^2;
        g_phi4_b(w) = g_phi4_b(w) + abs(a_phi_ab'*zb_3(:,x)).^2;
        g_phi4_c(w) = g_phi4_c(w) + abs(a_phi_cd'*zc_3(:,x)).^2;
        g_phi4_d(w) = g_phi4_d(w) + abs(a_phi_cd'*zd_3(:,x)).^2;
    end
end
g_phi4_a = 1/length(za_3).*g_phi4_a;
g_phi4_b = 1/length(zb_3).*g_phi4_b;
g_phi4_c = 1/length(zc_3).*g_phi4_c;
g_phi4_d = 1/length(zd_3).*g_phi4_d;
figure
plot(rad2deg(phi),g_phi4_a,'LineWidth',1.5), title('g(\phi) vs \phi for MIMO system ...
    (effect of PA non-linearity)'), grid minor, xlabel('\phi'), ylabel('Angular ...
    response');
hold on
plot(rad2deg(phi),g_phi4_b,'-.','LineWidth',1.5);
plot(rad2deg(phi),g_phi4_c,'LineWidth',1.5);
plot(rad2deg(phi),g_phi4_d,'-.','LineWidth',1.5);
legend('[M,\beta]=[8,(1,0)]', '[M,\beta]=[8,(1,-133)]', '[M,\beta]=[32,(1,0)]', ...
    '[M,\beta]=[32,(1,-133)]');
hold off
axes('Position',[0.2 0.4 .3 .4])
plot(rad2deg(phi),g_phi4_a,'LineWidth',1.5), grid minor, xlabel('\phi'), ...
    ylabel('Angular response');
hold on
plot(rad2deg(phi),g_phi4_b,'-.','LineWidth',1.5);
plot(rad2deg(phi),g_phi4_c,'LineWidth',1.5);
plot(rad2deg(phi),g_phi4_d,'-.','LineWidth',1.5);
hold off
xlim([39 41])
saveas(gcf,'P3_Q4','epsc');
```

1 Problem 1: Antenna Spacing and Beamforming

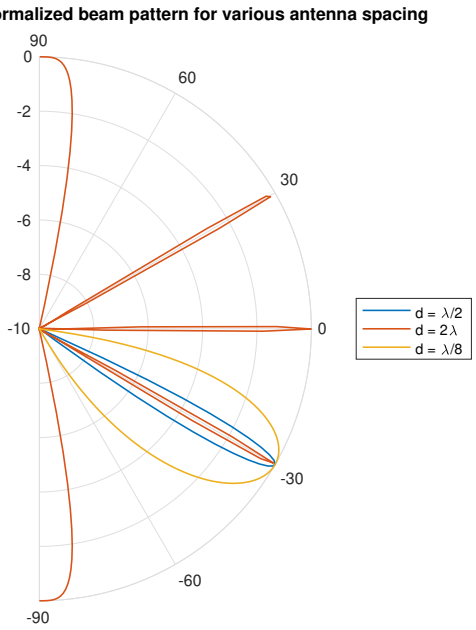

Figure 1: Normalized beam pattern for critically, sparsely and densely spaced antennae.

As antenna spacing deviates from being critically spaced ($d = \lambda/2$), beamforming ability of the antenna array in the desired direction starts to degrade. The equation for finding the main lobe angle ϕ is given by:

$$\cos(\phi) = \frac{k}{L} \text{mod} \frac{1}{\Delta}, k = (1, N-1) \tag{1}$$

Δ is the normalized antenna spacing and L is such that each element $e^{-j2\pi(N-1)\frac{d}{\lambda}\sin(\phi)}$ in $f(\phi)$ is equal to $e^{\frac{N-1}{L}}$.

- For critically spaced antennae, each basis vector in $f(\phi)$ has a single pair of main lobes (unique solution for (1), i.e. one main lobe and its mirror image), yielding in 1-1 mapping between angular windows and basis vectors.
- For sparsely spaced antennae, the basis vectors may have more than one pair of main lobes, which is why we observe several other lobes in Figure 1 for $d = 2\lambda$ (multiple solutions for (1)).
- For densely spaced antennae, some of the basis vectors have no main lobes (no solution for (1)), which explains the bulging of beam pattern in Figure 1 for $d = \lambda/8$.

2 Problem 2: Channel Capacities and Achievable Rates

Question 1: Channel Capacity

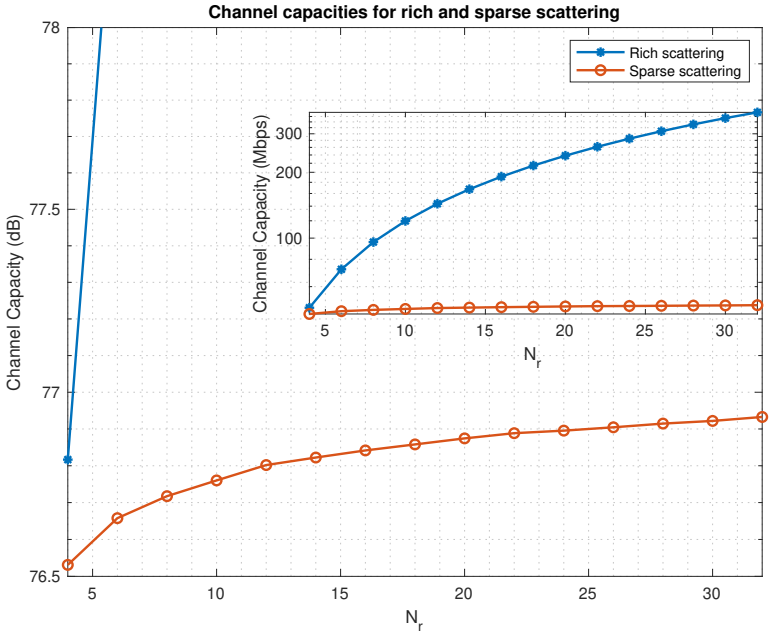

Figure 2: Channel capacities (dB) for rich and sparse scattering channels, zoomed in to showcase change of sparse scattering channel capacity with N_r. Inset shows the channel capacities (Mbps) in full scale for both channels.

Channel capacity for rich scattering is much larger than sparse scattering and channel capacity scales linearly with number of antennae for rich scattering. This is because a rich channel has significant number of multipath components due to multiple scatterers, leading to an increase in the rank of the channel matrix $\mathbf{H}$, allowing the MIMO system to exploit spatial diversity (increased spatial degrees of freedom) and improve spectral efficiency. The channel matrix is well conditioned in rich scattering environment, i.e., it has normalized i.i.d Gaussian entries and is usually assumed to have full rank.

In sparse scattering, the channel deviates from Gaussian assumptions due to fading correlation caused by few dominant scatterers (limited number of multipath components) and small angular spread, leading to smaller channel capacity from limited degrees of freedom.

Question 2: Achievable Rate

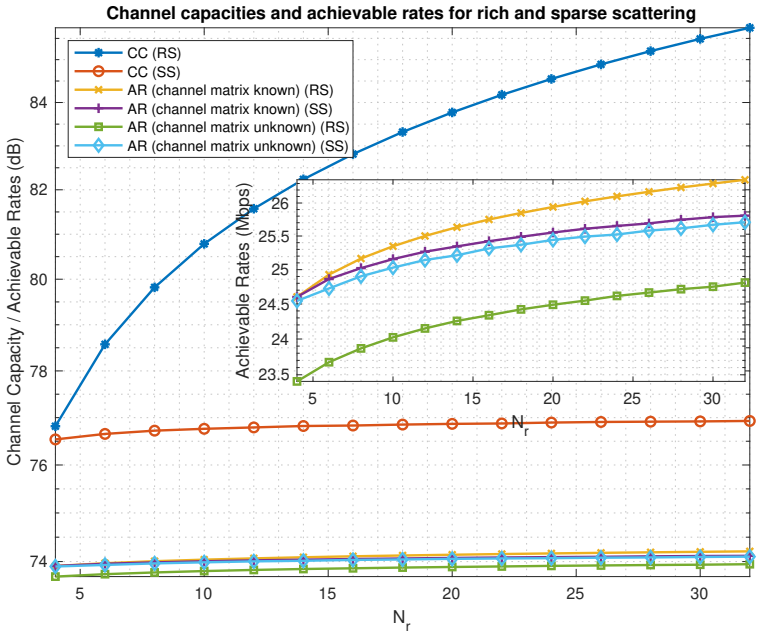

Figure 3: Channel capacities (CC) and achievable rates (AR) (dB) for rich (RS) and sparse scattering channels (SS) with known and unknown channel matrices. Inset shows the AR zoomed in for showcasing change with N_r in Mbps.

The highest achievable data rate is obtained through rich channel if channel matrix is known. For unknown channel matrix, the highest data is achieved by sparse channel. For known channel matrix, we do not observe significant data rate differences between rich and sparse channels (maximum difference: ~ 0.5 Mbps at $N_r = 32$). For beam training, there is an average difference of 1 Mbps for all values of N_r between rich and sparse capacities.

Achievable rates are lower than theoretical channel capacity because the channel rank K used, i.e. P is not full (P is limited to 2 in this case). This limit in the degree of freedom (i.e. multipath components) reduces achievable channel capacity. We have explained in Question 1 why a higher channel rank leads to higher capacity.

Beam training achieves lower data rates than SVD. This is expected because the SVD decomposition of the signal into precoding, combining and channel matrix selects the highest singular values (the diagonal matrix of SVD of H has descending diagonal elements) for communication, leading to higher achievable data rates compared to beam training. On the other hand, the angles selected from the angular space in beam training may yield the highest received signal power but still be suboptimal (and approximate) compared to SVD decomposition, leading to lower data rates in beam training.

We observe a discrepancy in beam training, i.e. achievable rate of sparse scattering is not lower but higher than rich scattering channel. We have explained in Question 1 why rich scattering theoretically should provide greater data rates than sparse scattering. However, beam training is essentially an approximation to the SVD method. Since we are already limiting the channel rank to 2, the benefits of having a rich channel is dominated by the error in channel estimation. However, sparse channels already have a limited number of multipath components to begin with, so the approximation in beam training does not hurt the data rates as badly as in rich scattering.

The phenomenon described above can also explain why the achievable rates for rich scattering and sparse scattering are so close when channel matrix is known compared to theoretical capacity. The bounded channel rank causes the data rates of rich scattering to approach the data rates of sparse scattering.

3 Problem 3: PA Non-Linearity and DAC Quantization

Question 1: Impact of DAC Quantization in the Frequency Domain

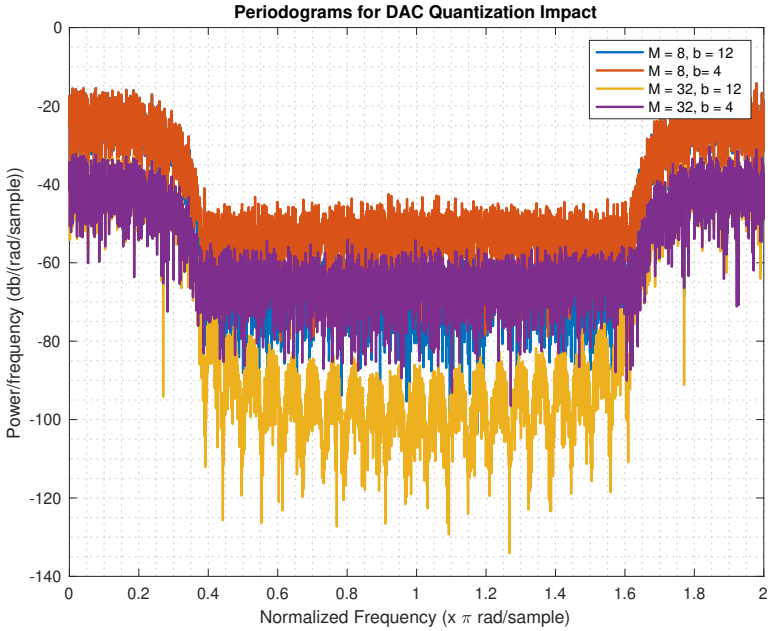

Figure 4: PSD of $z_1(n)$ under 4 cases for DAC quantization.

Maximum spectral regrowth occurs for M = 8. b = 4 and minimum spectral regrowth occurs for M = 32, b = 12. This is because

- Increasing the number of DAC bits reduces quantization noise.
- Increasing the number of antenna increases the probability of detecting the correct symbols as spatial multiplexing (hence reduce fading correlation and increase the number of independent data streams) is exploited to a higher extent.

Question 2: Impact of DAC Quantization in the Angular Domain

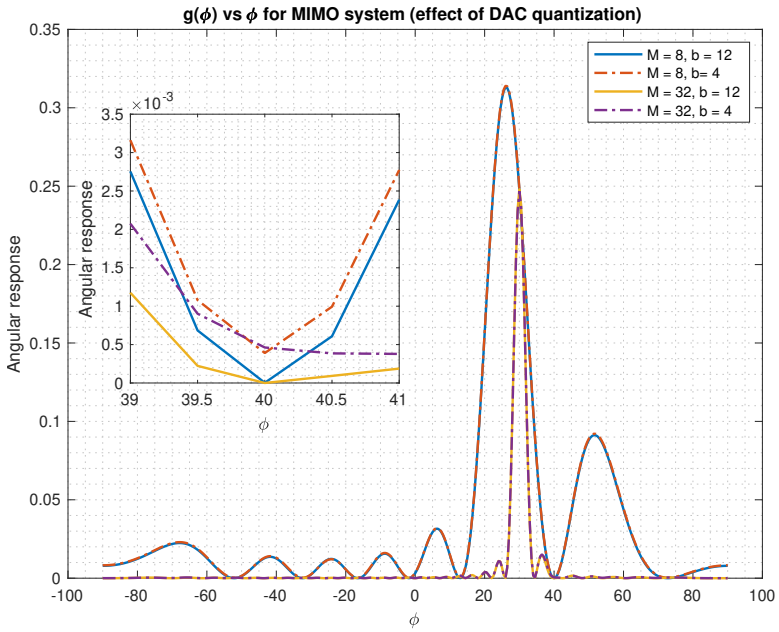

Figure 5: Plot of $g(\phi)$ vs ϕ under 4 cases for DAC quantization

Zero-forcing precoder allows directional communication, steering the 4-QAM bits towards UE-1 and zeros towards UE-2. Theoretically, due to finite DAC resolution, UE-1 will see a weak angular response while UE-2 will observe interference.

From Figure 5, we observe near negligible interference for both $M = 8$ and $M = 32$, for both $b = 4$ and $b = 12$, i.e. $g(40) \sim 0$. However, if we zoom in at $\phi = 40°$, we see that the greatest interference occurs for $M = 32, b = 4$ and the least interference occurs for $M = 32, b = 12$. Increasing the number of quantization bits decreases quantization noise, reducing interference among users. However, we observe that quantization noise dominates the spatial multiplexing benefits from increasing the number of antenna, which explains why $M = 32, b = 4$ generates more interference than $M = 8, b = 12$, although the former has more antennae than latter.

Question 3: Impact of PA Non-Linearity in the Frequency Domain

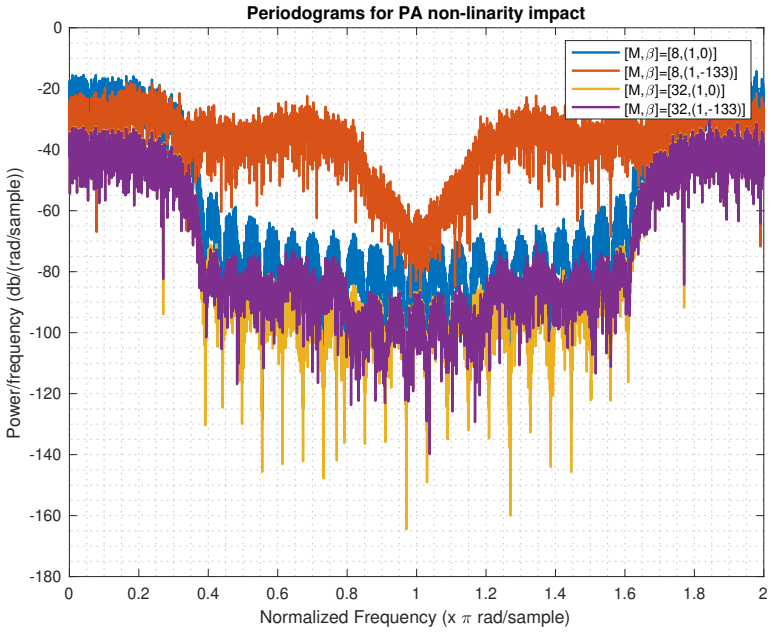

Figure 6: PSD of $z_1(n)$ under 4 cases for PA non-linearity.

Maximum spectral regrowth occurs for M = 8. β = 1, -133 and minimum spectral regrowth occurs for M = 32, β= 1, 0. This is expected because:

- If $\beta_3 \neq 0$, this means non-linearity is being introduced in the signal by the power amplifier, which results in out-of-band transmission due to intermodulation distortion.
- Increasing the number of antenna increases the probability of detecting the correct symbols as spatial multiplexing (hence reduce fading correlation and increase the number of independent data streams) is exploited to a higher extent.

Question 4: Impact of DAC Quantization in the Angular Domain

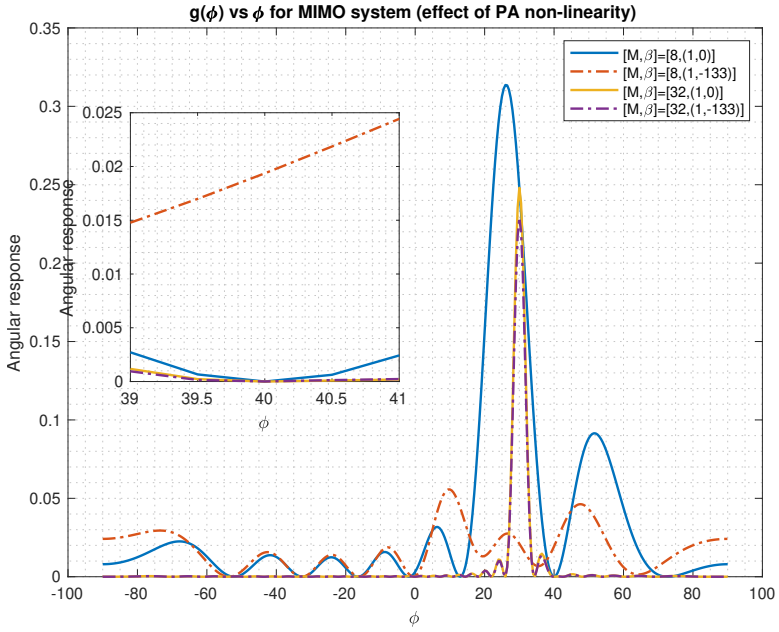

Figure 7: Plot of $g(\phi)$ vs ϕ under 4 cases for PA non-linearity.

Maximum interference occurs for M = 8, β = 1, -133 and minimum interference occurs for M = 32, β = 1, 0. This is because if $\beta_3 \neq 0$, the non-linearity introduced by the power amplifier increases intermodulation distortion, causing interference among adjacent users. Furthermore, if the number of antennae is reduced, then the user loses the benefits of spatial multiplexing. Similar to Question 2, we observe a dominance of PA non-linearity over the gains of spatial diversity, which explains why the curve for M = 8, β = 1, 0 is close to the curves M = 32, β = 1, 0 and M = 32, β = 1, -133.