ECE 233 - Spring 2020

Wireless Communications System Design, Modeling, and Implementation

Assignment 1

By: Swapnil Sayan Saha

UID: 605353215

Date of Submission: 2nd 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/14z5Emr5XPdd-7dDXHOr1VzWPNFa0TG5z/view?usp=sharing

```
clear; clc; close all;

%------------------------------------------------------------------------
fs = 20e6;       %Sampling rate (Hz)
Ts = 1/fs;       %Sampling time (s)
N = 64;          %No. of sub-carriers
delta_f = fs/N;  %Sub-carrier spacing (Hz)
%------------------------------------------------------------------------


%% Part 1: Generate STF

j= sqrt(-1);
S_26_26 = sqrt(1/2)* [0,0,1+j, ...
    0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0,0, ...
            0,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0,0,1+j,0,0,0,1+j,0,0,0,1+j,0,0];
x_stf = zeros(1,160); % Time domain signal (samples)
for n=0:159
    temp_sum = 0;
    for k=-26:26
        temp_sum = temp_sum + S_26_26(k+27)*exp(1j*2*pi*k*delta_f*n*Ts);
    end
    x_stf(n+1) = sqrt(1/12)* temp_sum;
end

figure, plot(1:160, real(x_stf)), title('STF - Real'), xlim([1 160])
saveas(gcf,'STF_plot','epsc'); %eps offers much higher resolution than jpg

%% Part 1: Generate LTF

L_26_26 = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,0, ...
           1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1];
x_ltf =zeros(1,160);
for n=0:159
    temp_sum = 0;
    for k=-26:26
        temp_sum = temp_sum + L_26_26(k+27)*exp(1j*2*pi*k*delta_f*n*Ts);
    end
    x_ltf(n+1) = sqrt(1/52)* temp_sum;
```

```
end

figure, plot(1:160, real(x_ltf)), title('LTF - Real'), xlim([1 160])
saveas(gcf,'LTF_plot','epsc');
%% Part 1: Packet with x_stf and x_ltf
x = [x_stf, x_ltf];

%% Part 2(a): Extract repeating sequence and magnitude plot of z_stf

z_stf = x(:,1:16); %repeating sequence
figure, plot(0:15,abs(z_stf),'LineWidth',1.5,'Marker','*'), title('|z_{stf}| vs ...
    n'), grid minor, xlabel('n'), ylabel('|z_{stf}|'); %|z_stf| vs n plot
saveas(gcf,'mag_zstf','epsc');
%% Part 2(b): Extract repeating sequence and magnitude plot of z_stf

figure, plot(0:15,unwrap(angle(z_stf)),'LineWidth',1.5,'Marker','*'), title('Phase ...
    z_{stf} (radian) vs n'), grid minor,xlabel('n'), ylabel('Phase z_{stf} ...
    (radian)'); %phase (radian) vs n plot
%The function unwrap removes the -pi to pi constraint in the angle()
%function. Using simply angle() function is also correct. An alternative is
%to use the phase() function, which will give answer in degrees.
saveas(gcf,'phase_zstf','epsc');
%% Part 2(c): Packet detection

num_trials = 10000; %10000 random trials
h = 1; %1 AWGN channel
sig_pow = rms(conv(x,h))^2; %calculate average complex signal power (unit: linear)
pack_detect = zeros(1,size(0:2:20,2)); %matrix to store probablities of packet ...
    detection for each SNR
m = 1;
for i = 0:2:20 %11 SNR values from 0 to 20 db
    noise_p = sig_pow/(db2mag(i))^2; %calculate noise power for a given SNR
    for j = 1:num_trials %10000 trials per SNR
        w_n = wgn(1,size(x,2),noise_p,'complex','linear');  %generate complex white ...
            gaussian noise
        y = conv(x,h) + w_n; %signal at receiver
        [r_n, lags] = xcorr(conj(y), z_stf,160); %cross correlation, maximum lag: 160.
        peak_cond = mean(abs(r_n))+(2.*std(abs(r_n))); %condition to denote r_n ...
            sample as peak
        [~,loc] = findpeaks(abs(r_n),lags,'MinPeakHeight',peak_cond); %find ...
            location of peaks
        if(sum(diff(loc)==16) >= 9-1) %denote packet is detected if 9 peaks are ...
            detected in r_n separated by 16 samples
        %The condition states 9-1 and not 9, because diff() will give the
        %difference between adjacent samples in location vector
        %Thus, you want to detect at least 9 peaks and you want them to be
        %separated by 16 samples, you will get 8 diff outputs, which
        %denotes we have 9 peaks.
            pack_detect(1,m) = pack_detect(1,m) + 1;
        end
    end
    pack_detect(1,m) = pack_detect(1,m)/num_trials; %compute probability
    m = m+1;
end
figure, semilogy(0:2:20,pack_detect,'LineWidth',1.5,'Marker','*'), title('Packet ...
    detection probability vs SNR'), grid minor, xlabel('SNR - dB'), ylabel('Packet ...
    Detection Probability'); %PDP vs SNR plot
saveas(gcf,'pack_detect','epsc');
%%  Part 3(a): Plotting mean of NSEE vs. SNR

num_trials = 10000; %10000 random trials
h = [1, 0.9, 0.5]; %AWGN channel with three multipaths.
sig_pow = rms(conv(x,h))^2; %calculate average complex signal power (unit: linear)
H_k_true = fft(h,64); %true value of channel coefficients
H_10 = H_k_true(1+10); %true H_10 value (+1 as FFT in matlab starts from 0)
H_22 = H_k_true(1+22); %true H_22 value (+1 as FFT in matlab starts from 0)
L_10 = L_26_26(27+10); %true value of L_10
L_22 = L_26_26(27+22); %true value of L_22
M_NSSE_10 = zeros(1,size(0:2:20,2)); %matrix to store mean of NSSE for k = 10 for ...
    each SNR
M_NSSE_22 = zeros(1,size(0:2:20,2)); %matrix to store mean of NSSE for k = 22 for ...
    each SNR
M_est_H_10_allTrials = []; %matrix to store estimated value of H_10 for all 10k ...
    trials for all SNR
M_est_H_22_allTrials = []; %matrix to store estimated value of H_10 for all 10k ...
    trials for all SNR
m = 1;
for i = 0:2:20 %11 SNR values from 0 to 20 db
    NSEE_10 = zeros(1,num_trials); %vector to store NSSE of 10000 trials, k = 10, ...
        one SNR
    NSEE_22 = zeros(1,num_trials); %vector to store NSSE of 10000 trials, k = 22, ...
        one SNR
    M_H_hat_10 = zeros(1,size(0:2:20,2)); %matrix to store estimated values of H_10 ...
        of 10000 trials, k = 10, one SNR
    M_H_hat_22 = zeros(1,size(0:2:20,2)); %matrix to store estimated values of H_22 ...
        of 10000 trials, k = 22, one SNR
    noise_p = sig_pow/(db2mag(i))^2; %calculate noise power for a given SNR
    for j = 1:num_trials %10000 trials per SNR
         w_n = wgn(1,size(x,2)+2,noise_p,'complex','linear');  %generate complex ...
             white gaussian noise, +2 to match conv dimension
         y = conv(x,h) + w_n; %signal at receiver
         y = y(1:end-2);
         y_ltf = y(161:end); % extract LTF from received signal
         v_n = y_ltf(33:96); %extract repeating LTF sequence
         L_hat_10 = 0; %scalar to store predicted L_10
         L_hat_22 = 0; %scalar to store predicted L_22
         %calculate predicted LTF symbol
         for n = 0:63
             L_hat_10 = L_hat_10 + (v_n(n+1)*exp(-1i*2*pi*10*delta_f*n*Ts));
             L_hat_22 = L_hat_22 + (v_n(n+1)*exp(-1i*2*pi*22*delta_f*n*Ts));
         end
         L_hat_10 = (1/8).*L_hat_10; %predicted L_10
         L_hat_22 = (1/8).*L_hat_22; %predicted L_22
         M_H_hat_10(1,j) = sqrt(52/64).*(L_hat_10 / L_10); %predicted H_10
         M_H_hat_22(1,j) = sqrt(52/64).*(L_hat_22 / L_22); %predicted H_22
         NSEE_10(1,j) = (abs(M_H_hat_10(1,j) - H_10))^2/(abs(H_10))^2; %NSSE of k = 10
         NSEE_22(1,j) = (abs(M_H_hat_22(1,j) - H_22))^2/(abs(H_22))^2; %NSSE of k = 22
    end
    M_NSSE_10(m) = mean(NSEE_10);
    M_NSSE_22(m) = mean(NSEE_22);
    M_est_H_10_allTrials = [M_est_H_10_allTrials; M_H_hat_10]; %appending NSEE for ...
        10k trials for k = 10 (for part 3c and part 4)
    M_est_H_22_allTrials = [M_est_H_22_allTrials; M_H_hat_22]; %appending NSEE for ...
        10k trials for k = 22 (for part 3c and part 4)
    m = m+1;
end
figure, plot(0:2:20,M_NSSE_10,'LineWidth',1.5,'Marker','*'), title('Mean of ...
    normalized squared estimation error, k = 10 and 22'), grid minor, xlabel('SNR - ...
    dB'), ylabel('Mean NSSE');
hold on;
plot(0:2:20,M_NSSE_22,'LineWidth',1.5,'Marker','*');
legend('k = 10', 'k = 22');
hold off;
saveas(gcf,'3_a','epsc');
clearvars -except H_10 H_22 delta_f fs h H_k_true M_est_H_10 M_est_H_22 N ...
    M_est_H_10_allTrials M_est_H_22_allTrials Ts x x_ltf x_stf z_stf
%% Part 3(b): Evaluating which subcarrier is better

%plot abs. value of FFT of h
figure, plot(0:63,abs(H_k_true),'LineWidth',1.5), title('Absolute value of 64-point ...
    FFT of h'), grid minor, xlim([0,63]);
hold on;
scatter(9,abs(H_k_true(10)),'b*')
text(9,abs(H_k_true(10)),'      k=10');
scatter(21,abs(H_k_true(22)),'r*')
text(21,abs(H_k_true(22)),'      k=22');
hold off;
saveas(gcf,'3_b','epsc');
%% Part 3(c) SER vs SNR

ABset = [-1/sqrt(2), 1/sqrt(2)]; %possible A, B values for X_k (4QAM)
```

```
nsym        = 12; % number of symbols
len_fft     = 64;   % 64 point fft
num_trials = 10000; %10000 random trials
m = 1;
BER_10 = zeros(size(0:2:20,2),num_trials); %matrix to store BER for 10th subcarrier ...
    for 10k trials
BER_22 = zeros(size(0:2:20,2),num_trials); %matrix to store BER for 22nd subcarrier ...
    for 10k trials
for i = 0:2:20 %11 SNR values from 0 to 20 db
    parfor j = 1:num_trials %parallel for loop to speed things up
        mod_data = zeros(nsym*N,1);
        for t = 1:nsym*N
            A_B =  datasample(ABset, 2); %randomly sample from set
            mod_data(t,1) = A_B(1) + (1i*A_B(2)); %generate 4QAM symbol
        end
        par_mod_data = reshape(mod_data,nsym,N); %serial to parallel conversion
        sym_10_true = par_mod_data(:,10); %true symbols at 10th subcarrier
        sym_22_true = par_mod_data(:,22); %true symbols at 22nd subcarrier
        % 64 data samples per symbol, 12 OFDM symbols
        IFFT_data = sqrt(N).*ifft(par_mod_data, len_fft,2); %64 point IFFT of ...
            sequence (across columns or each OFDM symbol)
        cylic_add_data_4QAM = [IFFT_data(:,49:64) IFFT_data]; %Add cyclic prefix
        x_data_4QAM = reshape(cylic_add_data_4QAM',1,80*nsym); %parallel to serial ...
            conversion
        x_4QAM = x_data_4QAM; %data packet data bits (omitted x_stf and x_ltf due ...
            to power issues)
        sig_pow_4QAM = rms(conv(x_4QAM,h))^2; %calculate average complex signal ...
            power (unit: linear)
        noise_p_4QAM = sig_pow_4QAM/(db2mag(i))^2; %calculate noise power for a ...
            given SNR
        w_n_4QAM = wgn(1,size(x_4QAM,2)+2,noise_p_4QAM,'complex','linear');  ...
            %generate complex white gaussian noise
        y_4QAM = conv(x_4QAM,h) + w_n_4QAM; %signal at receiver
        y_4QAM = y_4QAM(1:end-2);
        y_data_par = reshape(y_4QAM,80,nsym)'; %serial to parallel conversion
        y_data_WO_cyclic = y_data_par(:,17:80); %Remove cyclic prefix
        y_FFT = fft(y_data_WO_cyclic,len_fft,2); %64 point FFT of sequence (across ...
            columns or each OFDM symbol)
        %estimated symbols at 10th and 22nd subcarriers
        sym_10 = y_FFT(:,10)./M_est_H_10_allTrials(m,j);
        sym_22 = y_FFT(:,22)./M_est_H_22_allTrials(m,j);
        Cor_10 = 0; %no. of correct symbols for 10th subcarrier
        Cor_22 = 0; %no. of correct symbols for 22nd subcarrier
        %find number of correct symbols (comparing quadrants)
        for u = 1:nsym
            if((sign(real(sym_10(u))) == sign(real(sym_10_true(u)))) && ...
                (sign(imag(sym_10(u))) == sign(imag(sym_10_true(u)))))
                Cor_10 = Cor_10 + 1;
            end
            if((sign(real(sym_22(u))) == sign(real(sym_22_true(u)))) && ...
                (sign(imag(sym_22(u))) == sign(imag(sym_22_true(u)))))
                Cor_22 = Cor_22 + 1;
            end
        end
        BER_10(m,j) = (nsym -Cor_10) / nsym ; %Compute BER for 10th subcarrier
        BER_22(m,j) = (nsym -Cor_22) / nsym ; %Compute BER for 22nd subcarrier
     end
     m = m+1;
end
mean_BER_10 = (mean(BER_10,2))';
mean_BER_22 = (mean(BER_22,2))';
figure, semilogy(0:2:20,mean_BER_10,'LineWidth',1.5,'Marker','*'), title('BER vs ...
    SNR, k = 10 and 22'), grid minor, xlabel('SNR - dB'), ylabel('Mean BER');
hold on;
semilogy(0:2:20,mean_BER_22,'LineWidth',1.5,'Marker','*');
legend('k = 10', 'k = 22');
hold off;
saveas(gcf,'3_c','epsc');
%% Part 4(a) Adaptive loading (same modulation scheme):

M = [2,4,16,64]; %Modulation sizes: 2QAM, 4QAM, 16QAM, 64QAM
SNR = [0,10,20]; %SNR in db
alpha = 0:0.01:1; %fraction
H_est_index = [1,6,11]; %indices of estimated values of H_10 and H_22 for 0,10 and ...
    20 db.
num_trials = 10000; %10000 random trials
est_R_s = zeros(size(alpha,2),size(SNR,2)); %matrix to store mean sum rate for all ...
    SNR (same modulation used)
m = 1;
for i = alpha
    R = zeros(num_trials,size(SNR,2)); %matrix to store sum rate for each SNR for ...
        10000 trials
    for j = 1:num_trials
        n_10 = log2(1.+(abs(M_est_H_10_allTrials(H_est_index,j)).^2 .* i .* ...
            (db2mag(SNR).^2)')); %estimate spectrial efficiency for k = 10 for each SNR
        n_22 = log2(1.+(abs(M_est_H_22_allTrials(H_est_index,j)).^2 .* (1-i) .* ...
            (db2mag(SNR).^2)')); %estimate spectrial efficiency for k = 22 for each SNR
        mod_cond = (bsxfun(@minus,repmat(n_10,1,4),repmat(log2(M)',1,3)')>=0) & ...
            (bsxfun(@minus,repmat(n_22,1,4),repmat(log2(M)',1,3)')>=0); %find if ...
            both spectral efficiencies are greater than log2(M) for selected M for ...
            each SNR
        selected_mod = ones(1,size(SNR,2));
        for o = 1:size(SNR,2)
            if(any(mod_cond(o,:)))
```

```
                selected_mod(o) = M(find(mod_cond(o,:),1,'last')); %select ...
                    modulation with maximum size for each SNR
                %since selected_mod is already initialized to 1 and changed only if
                %mod_cond is met, so we do not need to set modulation size for an
                %SNR to 1 if spectral efficiencies are less than log2(M).
            end
        end
        R(j,1:size(SNR,2)) = (1.5.*log2(selected_mod).*delta_f)./10^6; %calculate ...
            sum rate
     end
     est_R_s(m,1:size(SNR,2)) = mean(R); %estimate mean sum rate
     m = m+1;
end
figure, plot(alpha,est_R_s,'LineWidth',1.5), title('Mean sum rate versus alpha for ...
    various SNR (same modulation scheme across all carriers)'), grid minor, ...
    xlabel('alpha'), ylabel('Mean Sum Rate');
legend('0-dB','10 dB', '20 dB');
saveas(gcf,'4_a','epsc');
%% Part 4(b) Adaptive loading (different modulation scheme):

M = [2,4,16,64]; %Modulation sizes: 2QAM, 4QAM, 16QAM, 64QAM
SNR = [0,10,20]; %SNR in db
alpha = 0:0.01:1; %fraction
H_est_index = [1,6,11]; %indices of estimated values of H_10 and H_22 for 0,10 and ...
    20 db.
num_trials = 10000; %10000 random trials
est_R_d = zeros(size(alpha,2),size(SNR,2)); %matrix to store mean sum rate for all SNR
m = 1;
for i = alpha
    R = zeros(num_trials,size(SNR,2)); %matrix to store sum rate for each SNR for ...
        10000 trials
    for j = 1:num_trials
        n_10 = log2(1.+(abs(M_est_H_10_allTrials(H_est_index,j)).^2 .* i .* ...
            (db2mag(SNR).^2)')); %estimate spectrial efficiency for k = 10 for each SNR
        n_22 = log2(1.+(abs(M_est_H_22_allTrials(H_est_index,j)).^2 .* (1-i) .* ...
            (db2mag(SNR).^2)')); %estimate spectrial efficiency for k = 22 for each SNR
        %individual conditions now, instead of "and" condition
        mod_cond_10 = (bsxfun(@minus,repmat(n_10,1,4),repmat(log2(M)',1,3)')>=0);
        mod_cond_22 = (bsxfun(@minus,repmat(n_22,1,4),repmat(log2(M)',1,3)')>=0);
        selected_mod_10 = ones(1,size(SNR,2));
        selected_mod_22 = ones(1,size(SNR,2));
        for o = 1:size(SNR,2)
            if(any(mod_cond_10(o,:)))
                selected_mod_10(o) = M(find(mod_cond_10(o,:),1,'last')); %select ...
                    modulation with maximum size for each SNR
            end
            if(any(mod_cond_22(o,:)))
```

```
                selected_mod_22(o) = M(find(mod_cond_22(o,:),1,'last')); %select ...
                    modulation with maximum size for each SNR
            end
        end
        R(j,1:size(SNR,2)) = ...
            (0.75.*(log2(selected_mod_10)+log2(selected_mod_22)).*delta_f)./10^6; ...
            %calculate sum rate
    end
    est_R_d(m,1:size(SNR,2)) = mean(R); %estimate mean sum rate
    m = m+1;
end
figure, plot(alpha,est_R_d,'LineWidth',1.5), title('Mean sum rate versus alpha for ...
    various SNR (different modulation scheme across all carriers)'), grid minor, ...
    xlabel('alpha'), ylabel('Mean Sum Rate');
legend('0-dB','10 dB', '20 dB');
saveas(gcf,'4_b','epsc');
%% Find optimal alpha for SNR = 20 db for part 4(a) and 4(b)

[~,loc_20_s] = max(est_R_s(:,3));
alpha_20_s = alpha(loc_20_s);
[~,loc_20_d] = max(est_R_d(:,3));
alpha_20_d = alpha(loc_20_d);
```

1 Part 1: Packet Generation

Plot of STF and LTF samples

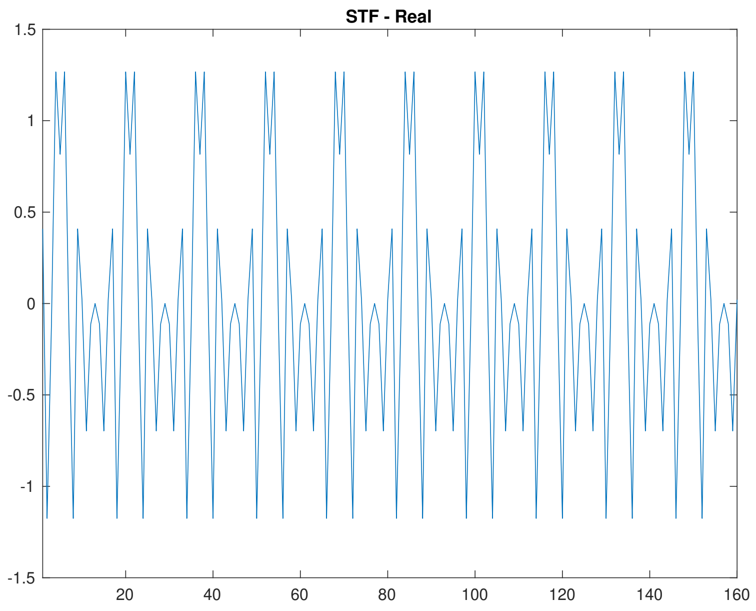

Figure 1: Plot of real part of $x_{\text{STF}}(n)$ vs. n

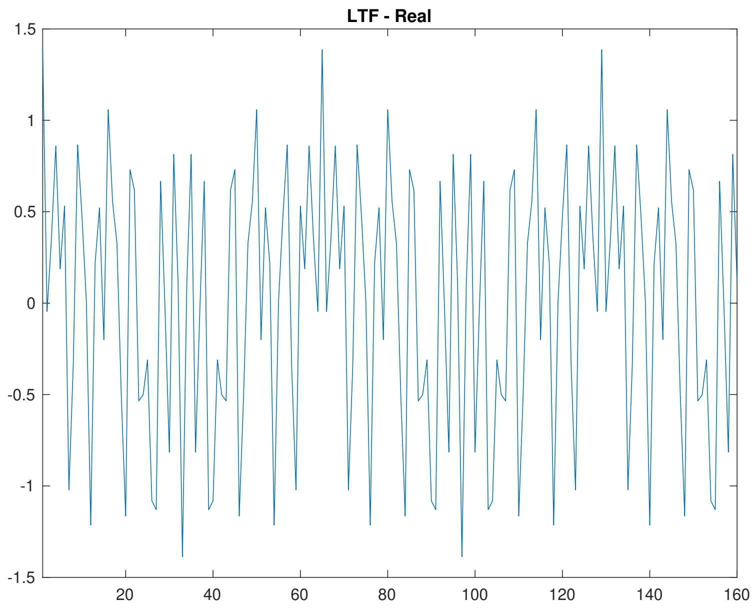

Figure 2: Plot of real part of $x_{\text{LTF}}(n)$ vs. n

2 Part 2: Packet Detection

a:

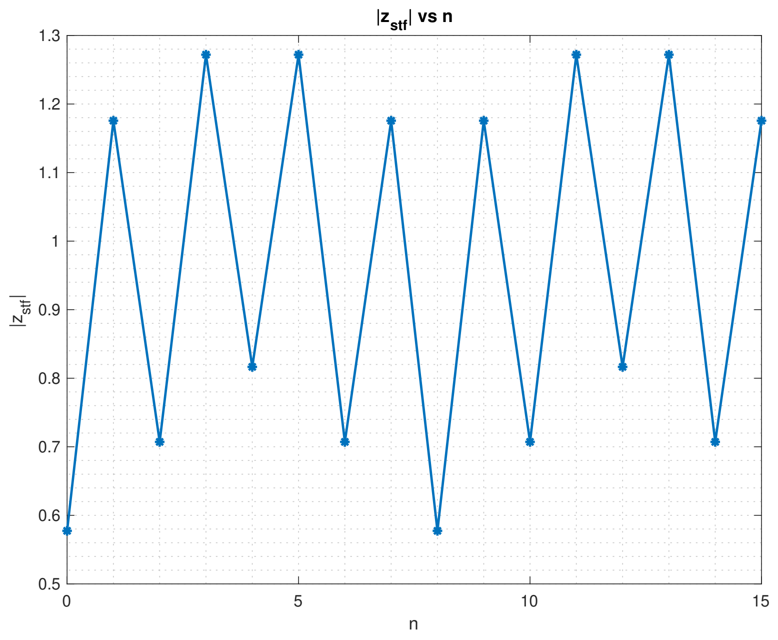

Figure 3: Plot of $|z_{\text{STF}}(n)|$ vs. n

b:

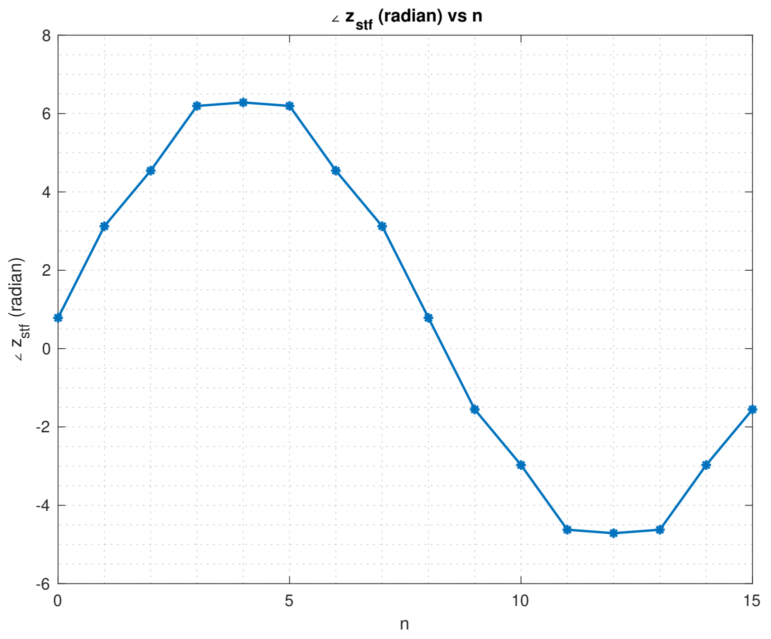

Figure 4: Plot of $\angle z_{\text{STF}}(n)$ vs. n

c:

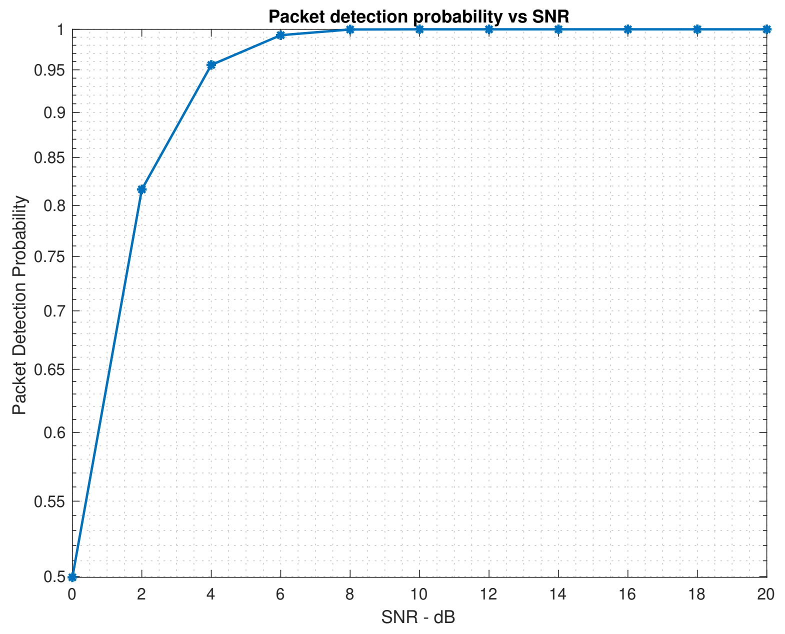

Figure 5: Plot of probability of packet detection vs. SNR

3 Part 3: Channel Estimation

a:

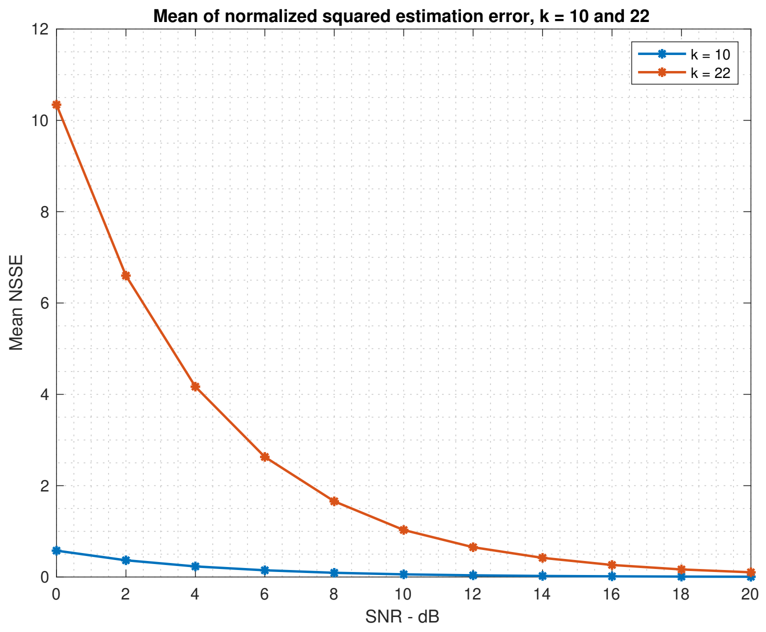

Figure 6: Plot of mean normalized squared channel estimation error versus the SNR for k = 10 and k = 22

b:

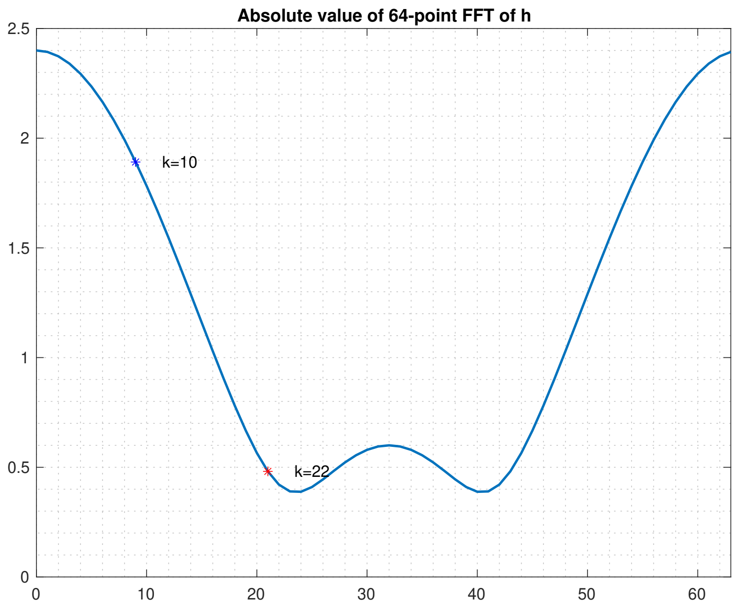

Figure 7: Plot of magnitude of 64-point FFT of h

From the FFT-plot in Figure 7, we can see that frequency selective fading is more prominent for subcarrier k = 22 than subcarrier k = 10. As a result subcarrier k = 22 has a higher channel estimation error due to higher data degradation. Frequency selective fading occurs when the coherence bandwidth of the channel is smaller than signal bandwidth.

c:

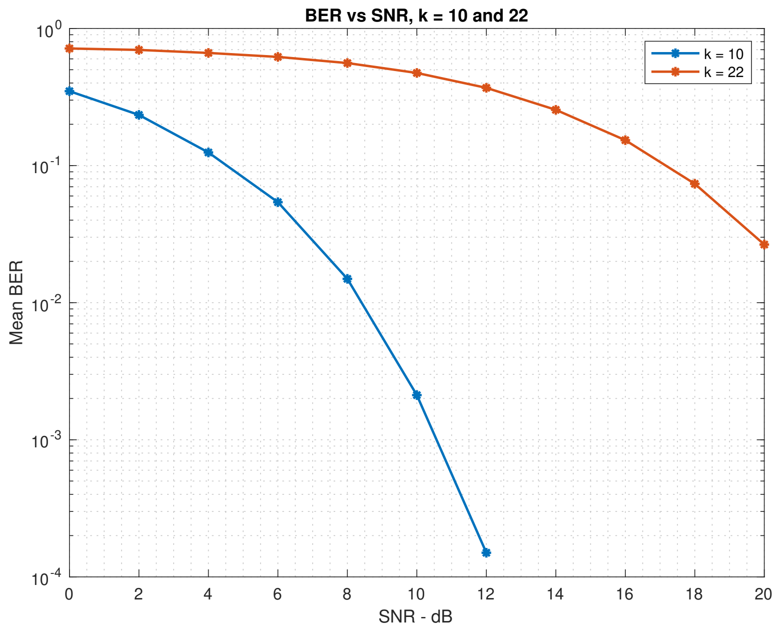

Figure 8: Plot of the symbol error rate versus the SNR

4 Part 4: Adaptive Loading and Power Allocation

a:

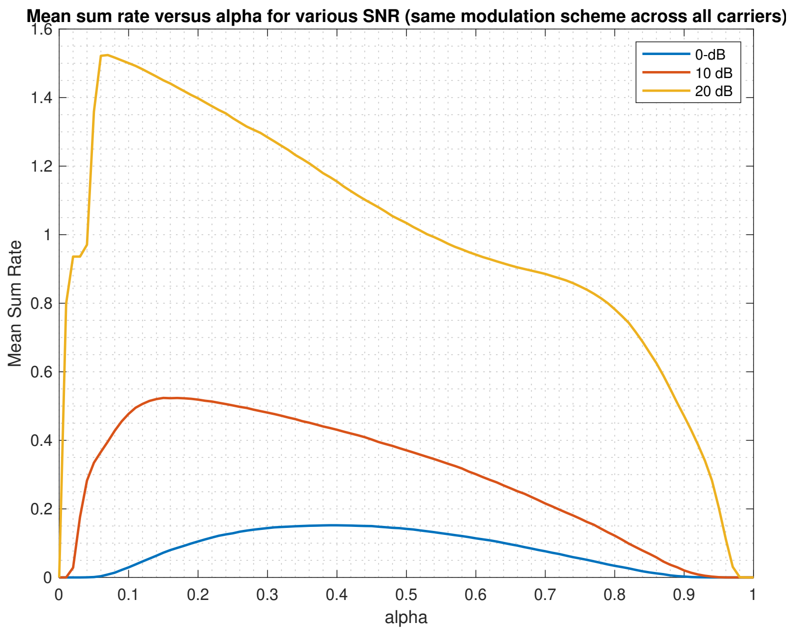

Figure 9: Plot of the mean sum rates versus the power fraction α when modulation level is the same

b:

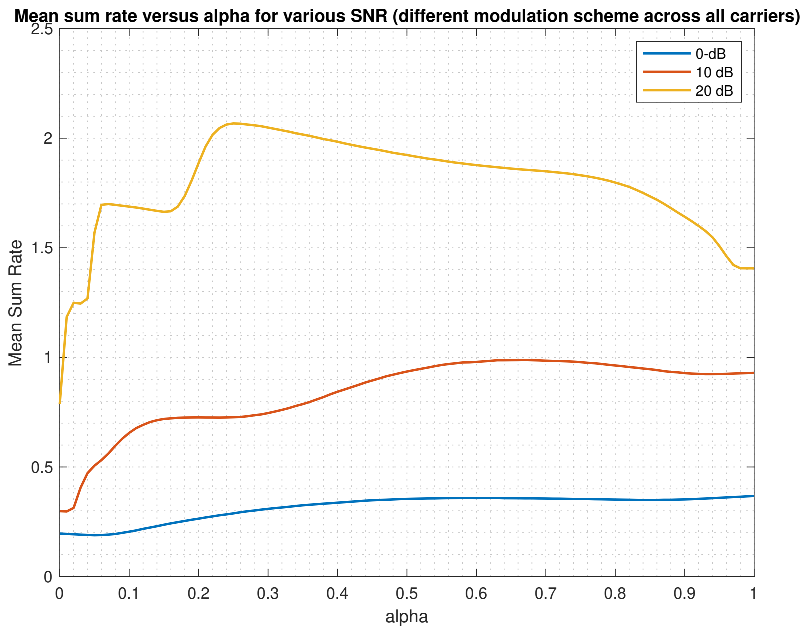

Figure 10: Plot of the mean sum rates versus the power fraction α when modulation level can be different

From Figure 9 and 10, we can see that that the mean sum rate for all SNR is higher when different modulation types can be used for each subcarrier compared to same modulation type. We can also observe that the mean sum rate yields a flat / consistent response over all values of alpha for each SNR when different modulation types can be used for each subcarrier compared to same modulation type. This is because in part (b), we are trying to maximize the channel capacity by loading more bits on the better subcarrier (in this case k = 10) through choosing a higher modulation scheme, yet ensuring that none of the subcarriers are wasted. This is called bit loading. In part (a) however, we are essentially doing power loading, i.e. allocating more power to better subcarrier. As a result, as power allocated to k = 10 increases, mean sum rate increases until an optimum point after which the mean sum rates starts to decrease despite allocating more power to better subcarrier as the same number of bits are being carried by all subcarriers here, so allocating less power to bad subcarriers translates to losing those bits. For this same reason, the optimal α for SNR = 20 dB is 0.07 for part (a) and 0.25 for part (b). For both plots, as the SNR increases, the mean sum rate increases.