

Shaira Alam

UID: 506302126

Homework 1

Dr. Danijela Cabric

ECE 233: Wireless Communications System Design, Modeling, and Implementation

Summer 2024

---

# Part 1: SSB Generation

## Table of Contents

Parameters .....	1
PSS .....	1
SSS .....	1
PBCH Pilot .....	2
PBCH Data .....	2
Concatenation .....	2
Channel and Noise .....	2
(a) $s(n)$ Transmitted Signal .....	3
(b) $y(n)$ Received Signal .....	3
Function Definitions .....	4

## Parameters

```
FFT_size = 4096;
CP_length = 288;
CP_OFDM_length = FFT_size+CP_length;
SCS = 30e3;
Ts = 1/FFT_size/SCS;
delta_f = 2.03e3; % kHz
```

```
QAM_mod = 4;
num_sc = 240;
N_id_1 = 77;
N_id_2 = 2;
c_init = 120897;
```

```
SNR = 20;
h = [0 0 0 0 1 0.5]';
```

## PSS

```
% OFDM Modulation
PSS_stream = generate_PSS_BPSK(N_id_2);
% Map symbol to subcarrier
d_PSS = [zeros(56,1);PSS_stream;zeros(FFT_size-183,1)];
% FFT
OFDM_PSS_body = ifft(d_PSS)*sqrt(FFT_size);
% Add CP
CP_OFDM_PSS = [OFDM_PSS_body(end-CP_length+1:end);OFDM_PSS_body];
```

## SSS

```
% OFDM Modulation
SSS_stream = generate_SSS_BPSK(N_id_1, N_id_2);
```

```
% Map symbol to subcarrier
d_SSS = [zeros(56,1);SSS_stream;zeros(FFT_size-183,1)];
% FFT
OFDM_SSS_body = ifft(d_SSS)*sqrt(FFT_size);
% Add CP
CP_OFDM_SSS = [OFDM_SSS_body(end-CP_length+1:end);OFDM_SSS_body];
```

## PBCH Pilot

```
% OFDM Modulation
PBCH_pilot_stream = generate_PBCH_pilot(c_init);
% Map symbol to subcarrier
d_PBCH_pilot = zeros(FFT_size,1);
k = 0:59;
d_PBCH_pilot(1+4*k+1) = PBCH_pilot_stream;
% FFT
OFDM_PBCH_pilot_body = ifft(d_PBCH_pilot)*sqrt(FFT_size);
% Add CP
CP_OFDM_PBCH_pilot = [OFDM_PBCH_pilot_body(end-
CP_length+1:end);OFDM_PBCH_pilot_body];
```

## PBCH Data

```
% OFDM Modulation
PBCH_data_stream = generate_PBCH_data(num_sc, QAM_mod);
% Map symbol to subcarrier
d_PBCH_data = [PBCH_data_stream;zeros(FFT_size-num_sc,1)];
% FFT
OFDM_PBCH_data_body = ifft(d_PBCH_data)*sqrt(FFT_size);
% Add CP
CP_OFDM_PBCH_data = [OFDM_PBCH_data_body(end-
CP_length+1:end);OFDM_PBCH_data_body];
```

## Concatenation

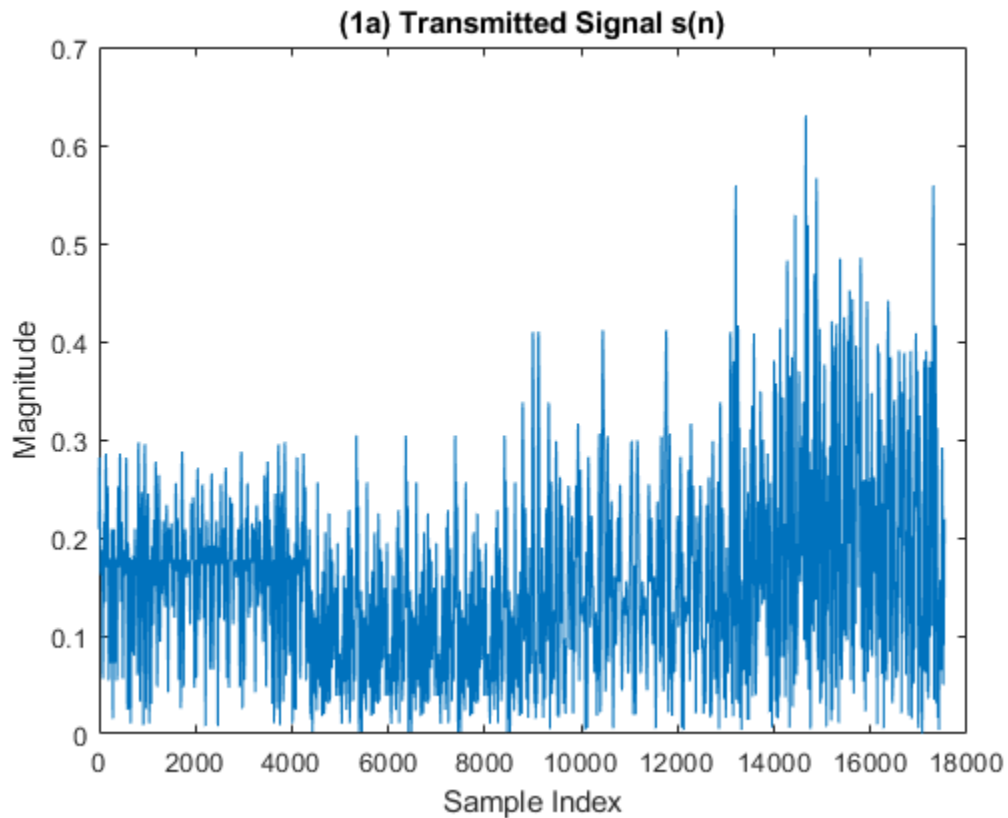
```
CP_OFDM_chain = [CP_OFDM_PSS; CP_OFDM_PBCH_pilot; CP_OFDM_SSS;
CP_OFDM_PBCH_data];
```

## Channel and Noise

```
signal_after_channel = conv(CP_OFDM_chain,h);
N_0 = 10^(-SNR/10) * (norm(signal_after_channel)^2/
length(signal_after_channel));
noise = sqrt(N_0/2)*(randn(length(signal_after_channel),1) +
1j*randn(length(signal_after_channel),1));
received_signal = signal_after_channel .*
exp(1j*2*pi*delta_f*(0:length(signal_after_channel)-1)*Ts) + noise;
```

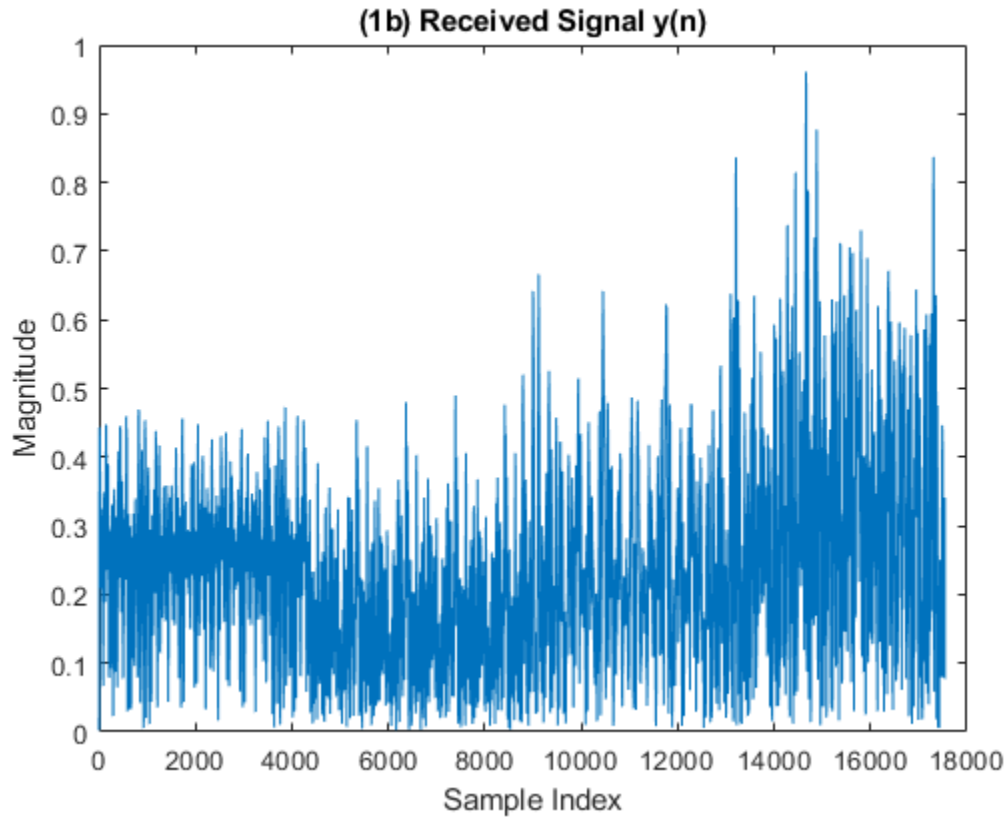
## (a) $s(n)$ Transmitted Signal

```
figure;  
plot(abs(CP_OFDM_chain));  
xlabel('Sample Index'); % X-axis label  
ylabel('Magnitude'); % Y-axis label  
title('(1a) Transmitted Signal s(n)');
```



## (b) $y(n)$ Received Signal

```
figure;  
plot(abs(received_signal));  
xlabel('Sample Index'); % X-axis label  
ylabel('Magnitude'); % Y-axis label  
title('(1b) Received Signal y(n)');
```



## Function Definitions

```
function PSS_BPSK = generate_PSS_BPSK(N_id_2)
    x = zeros(127,1);
    PSS_BPSK = zeros(127,1);
    x_init = [0 1 1 0 1 1 1];
    x(1:7) = x_init;
    for i = 1:120
        x(i+7) = mod(x(i+4)+x(i),2);
    end
    for n = 0:126
        m = mod(n + 43*N_id_2,127);
        PSS_BPSK(n+1) = 1-2*x(m+1);
    end
end

function SSS_BPSK = generate_SSS_BPSK(N_id_1,N_id_2)
    x_0 = zeros(127,1);
    x_1 = zeros(127,1);
    SSS_BPSK = zeros(127,1);
    x_init = [1 0 0 0 0 0 0];
    x_0(1:7) = x_init;
    x_1(1:7) = x_init;

    for i = 1:120
```

```
x_0(i+7) = mod(x_0(i+4)+x_0(i),2);
x_1(i+7) = mod(x_1(i+1)+x_1(i),2);
end
for n = 0:126
    m_0 = mod(n + 15* floor(N_id_1/112) + 5*N_id_2,127);
    m_1 = mod(n + mod(N_id_1,112),127);
    SSS_BPSK(n+1) = (1-2*x_0(m_0+1))*(1-2*x_1(m_1+1));
end
end

function QPSK_pilot_stream = generate_PBCH_pilot(c_init)
    c = zeros(120,1);
    QPSK_pilot_stream = zeros(60,1);
    x_1 = zeros(1800,1);
    x_2 = zeros(1800,1);
    x_1_init = [1; zeros(30,1)];
    x_1(1:31) = x_1_init;
    x_2_init = zeros(31,1);
    x_2_init_char = dec2bin(c_init);
    for i = 1:length(x_2_init_char)
        x_2_init(length(x_2_init_char)-i+1) = str2double(x_2_init_char(i));
    end
    x_2(1:31) = x_2_init;
    for n = 1:1800
        x_1(n+31) = mod(x_1(n+3)+x_1(n),2);
        x_2(n+31) = mod(x_2(n+3)+x_2(n+2)+x_2(n+1)+x_2(n),2);
    end
    for n = 0:119
        c(n+1) = mod(x_1(n+1600+1)+x_2(n+1600+1),2);
    end
    for n = 0:59
        QPSK_pilot_stream(n+1) = 1/sqrt(2)*(1-2*c(2*n+1)) + 1j/
sqrt(2)*(1-2*c(2*n+2));
    end
end

function QPSK_data_stream = generate_PBCH_data(num_sc, QAM_mod)
    data_bit_stream = randi([0 1],num_sc*log2(QAM_mod),1);
    QPSK_data_stream =
qammod(data_bit_stream,QAM_mod,InputType='bit',UnitAveragePower=true);
end
```

*Published with MATLAB® R2024a*

---

# Part 2: Frequency offset compensation & PSS search

## Table of Contents

Parameters .....	1
PSS .....	1
Channel and Noise .....	2
Coarse frequency offset estimation .....	2
Fine Frequency Offset Estimation .....	2
(a) Determine (NID2, Frequency offset estimation) .....	3
Calculate root mean square .....	3
Fine Frequency Offset Estimation .....	3
Calculate RMSE .....	4
(b) Plot RMSE vs SNR .....	4
Function Definitions .....	4

## Parameters

```
FFT_size = 4096;
CP_length = 288;
SCS = 30e3;
Ts = 1/FFT_size/SCS;
CP_OFDM_length = FFT_size+CP_length;
num_sc = 240; % number of subcarriers
N_id_2 = 2
delta_f = 2.03e3; % kHz
```

```
N_id_2 =

     2
```

## PSS

```
% OFDM Modulation
PSS_stream = PSS_BPSK(N_id_2);
% Map symbol to subcarrier
d_PSS = [zeros(56,1);PSS_stream;zeros(FFT_size-183,1)];
% FFT
OFDM_PSS_body = ifft(d_PSS)*sqrt(FFT_size);
% Add CP
CP_OFDM_PSS = [OFDM_PSS_body(end-CP_length+1:end);OFDM_PSS_body];
```

## Channel and Noise

```
h = 1;
signal_after_channel = conv(CP_OFDM_PSS,h);

SNR = 20;
N_0 = 10^(-SNR/10) * (norm(signal_after_channel)^2/
length(signal_after_channel));
noise = sqrt(N_0/2)*(randn(length(signal_after_channel),1) +
1j*randn(length(signal_after_channel),1));
received_PSS_signal = signal_after_channel .*
exp(1j*2*pi*delta_f*(0:length(signal_after_channel)-1)*Ts) + noise;
```

## Coarse frequency offset estimation

```
% Searches from -15kHz to 15kHz with 100 Hz incremenets
freq_offset = -SCS/2:100:SCS/2;

% Correlation results between received signal and reference signals for each
value of NID2: 3 row matrix (0-2) for NID2 corresponding frequency offsets
corr = zeros(3,length(freq_offset));

for i = 0:2 % iterates through each NID2 value
    PSS_ref_stream = PSS_BPSK(i); % reference waveform for NID2 value
    d_PSS_ref = [zeros(56,1);PSS_ref_stream;zeros(FFT_size-183,1)]; %
prepares for modulation
    OFDM_PSS_ref_body = ifft(d_PSS_ref); % converts to frequency domain
    CP_OFDM_PSS_ref = [OFDM_PSS_ref_body(end-
CP_length+1:end);OFDM_PSS_ref_body]; % adds CP to OFDM signal

    for j = 1:numel(freq_offset) % iterates thorough each frequency offset
value to compute correlation for different frequency offsets
        % Calculates correlation between received signal and reference
signal with frequency offset
        corr(i+1,j) = abs(received_PSS_signal' * (CP_OFDM_PSS_ref .*
exp(1j*2*pi*freq_offset(j)*(0:length(CP_OFDM_PSS_ref)-1)*Ts)));
    end
end

% Finds position of maximum correlation value
[~,N_id_2_est_pos] = max(max(abs(corr),[],2));
% converts index position to actual NID2 value
N_id_2_est = N_id_2_est_pos - 1;
```

## Fine Frequency Offset Estimation

```
freq_offset_est = -angle(received_PSS_signal(FFT_size+1:FFT_size+CP_length) '
* received_PSS_signal(1:CP_length))/2/pi*SCS;
freq_offset_est_kHz = freq_offset_est / 1e3;
```



## (a) Determine (NID2, Frequency offset estimation)

```
disp('(2a) Determine N(2)_ID and Estimated Frequency Offset: ');
disp(['Estimated N(2)_ID: ', num2str(N_id_2_est)]); % from course frequency
offset estimation
disp(['Estimated Frequency Offset (Hz): ', num2str(freq_offset_est)]); %
from fine frequency offset estimation
disp(['Estimated Frequency Offset (kHz): ', num2str(freq_offset_est_kHz)]);
```

```
(2a) Determine N(2)_ID and Estimated Frequency Offset:
Estimated N(2)_ID: 2
Estimated Frequency Offset (Hz): 2043.1866
Estimated Frequency Offset (kHz): 2.0432
```

## Calculate root mean square

```
FFT_size = 4096;
CP_length = 288;
SCS = 30e3;
Ts = 1/FFT_size/SCS;
CP_OFDM_length = FFT_size + CP_length;
num_sc = 240; % number of subcarriers
N_id_2 = 2;
delta_f = 2.03e3; % kHz
num_MC = 1000;

SNR_range = 0:2:20; % SNR values in dB
num_SNR = length(SNR_range);
RMSE = zeros(num_MC, num_SNR); % Root Mean Square Error

for SNR_id = 1:num_SNR
    for MC_id = 1:num_MC

        SNR = SNR_range(SNR_id);
        N_0 = 10^(-SNR/10) * (norm(signal_after_channel)^2/
length(signal_after_channel));
        noise = sqrt(N_0/2)*(randn(length(signal_after_channel),1) +
1j*randn(length(signal_after_channel),1));
        received_PSS_signal = signal_after_channel .*
exp(1j*2*pi*delta_f*(0:length(signal_after_channel)-1)*Ts) + noise;
```

## Fine Frequency Offset Estimation

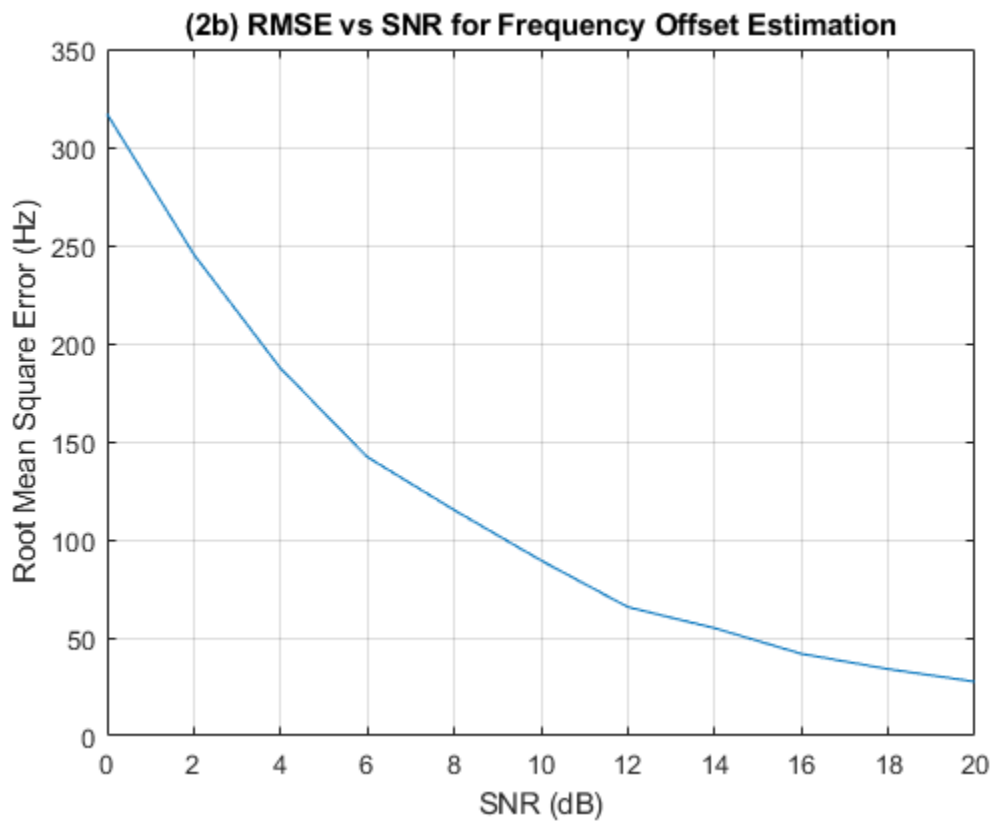
```
freq_offset_est =
-angle(received_PSS_signal(FFT_size+1:FFT_size+CP_length)' *
received_PSS_signal(1:CP_length))/2/pi*SCS;
freq_offset_est_kHz = freq_offset_est / 1e3;
```

## Calculate RMSE

```
RMSE(MC_id, SNR_id) = (freq_offset_est - delta_f)^2;  
  
end  
end  
  
mean_RMSE = sqrt(mean(RMSE, 1));
```

## (b) Plot RMSE vs SNR

```
figure;  
plot(SNR_range, mean_RMSE);  
xlabel('SNR (dB)');  
ylabel('Root Mean Square Error (Hz)');  
title('(2b) RMSE vs SNR for Frequency Offset Estimation');  
grid on;
```



## Function Definitions

```
function BPSK_stream = PSS_BPSK(N_id_2)  
    x = zeros(127,1);  
    BPSK_stream = zeros(127,1);  
    x_init = [0 1 1 0 1 1 1];
```

```
x(1:7) = x_init;  
for i = 1:120  
    x(i+7) = mod(x(i+4)+x(i),2);  
end  
for n = 0:126  
    m = mod(n + 43*N_id_2,127);  
    BPSK_stream(n+1) = 1-2*x(m+1);  
end  
end
```

*Published with MATLAB® R2024a*

---

# Part 3: Timing synchronization

## Table of Contents

Parameters .....	1
PSS and PBCH_pilot .....	1
Concatenation .....	1
Channel and Noise .....	2
Timing Synchronization .....	2
(a) Plot Correlation for all SNR values .....	2

## Parameters

```
FFT_size = 4096;
CP_length = 288;
SCS = 30e3;
Ts = 1/FFT_size/SCS;
CP_OFDM_length = FFT_size+CP_length;
num_sc = 240;
N_id_2 = 2;
c_init = 120897;
```

## PSS and PBCH\_pilot

```
% OFDM Modulation
PSS_stream = PSS_BPSK(N_id_2);
PBCH_pilot_stream = PBCH_QPSK(c_init);

% Map symbol to subcarrier
d_PSS = [zeros(56,1);PSS_stream;zeros(FFT_size-183,1)];
d_PBCH_pilot = zeros(FFT_size,1);
k = 0:59;
d_PBCH_pilot(1+4*k+1) = PBCH_pilot_stream(1:60);

% FFT
OFDM_PSS_body = ifft(d_PSS)*sqrt(FFT_size);
OFDM_PBCH_pilot_body = ifft(d_PBCH_pilot)*sqrt(FFT_size);

% Add CP
CP_OFDM_PSS = [OFDM_PSS_body(end-CP_length+1:end);OFDM_PSS_body];
CP_OFDM_PBCH_pilot = [OFDM_PBCH_pilot_body(end-CP_length+1:end);OFDM_PBCH_pilot_body];
```

## Concatenation

```
CP_OFDM_chain = [CP_OFDM_PSS;CP_OFDM_PBCH_pilot];
```

## Channel and Noise

```
h = [0 0 0 0 1 0.5]';
signal_after_channel = conv(CP_OFDM_chain,h);

SNR_values = [-5, 20]; % SNR values in dB
corr_all = zeros(length(SNR_values), FFT_size + 1);

for snr_idx = 1:length(SNR_values)

    SNR = SNR_values(snr_idx);
    N_0 = 10^(-SNR/10) * (norm(signal_after_channel)^2/
length(signal_after_channel));
    noise = sqrt(N_0/2)*(randn(length(signal_after_channel),1) +
1j*randn(length(signal_after_channel),1));
    received_signal = signal_after_channel + noise;
```

## Timing Synchronization

```
corr = zeros(FFT_size+1,1);
for m = 0:FFT_size
    corr(m+1) = abs(received_signal(m+1:m+CP_OFDM_length)' *
CP_OFDM_PSS);
end
[~,m_STO] = max(corr);
m_STO = m_STO - 1; % adjusting the index
disp(['mSTO for SNR = ', num2str(SNR), ' dB: ', num2str(m_STO)]);

corr_all(snr_idx, :) = corr;

mSTO for SNR = -5 dB: 4
mSTO for SNR = 20 dB: 4

end
```

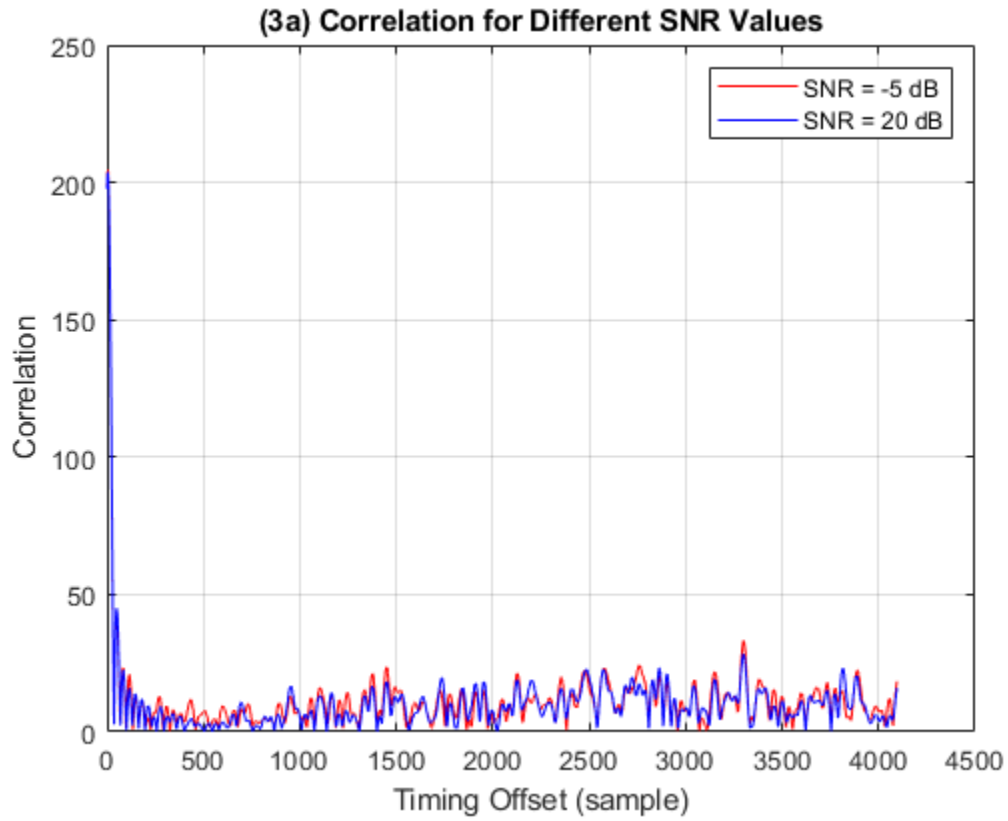
### (a) Plot Correlation for all SNR values

```
figure;
plot(0:FFT_size, corr_all(1, :), 'r', 'DisplayName', 'SNR = -5 dB');
hold on;
plot(0:FFT_size, corr_all(2, :), 'b', 'DisplayName', 'SNR = 20 dB');
xlabel("Timing Offset (sample)")
ylabel("Correlation")
title('(3a) Correlation for Different SNR Values')
legend show;
grid on;

function BPSK_stream = PSS_BPSK(N_id_2)
    x = zeros(127,1);
    BPSK_stream = zeros(127,1);
    x_init = [0 1 1 0 1 1 1];
```

```
x(1:7) = x_init;
for i = 1:120
    x(i+7) = mod(x(i+4)+x(i),2);
end
for n = 0:126
    m = mod(n + 43*N_id_2,127);
    BPSK_stream(n+1) = 1-2*x(m+1);
end
end

function QPSK_stream = PBCH_QPSK(c_init)
    c = zeros(120,1);
    QPSK_stream = zeros(60,1);
    x_1 = zeros(1800,1);
    x_2 = zeros(1800,1);
    x_1_init = [1; zeros(30,1)];
    x_1(1:31) = x_1_init;
    x_2_init = zeros(31,1);
    x_2_init_char = dec2bin(c_init);
    for i = 1:length(x_2_init_char)
        x_2_init(length(x_2_init_char)-i+1) = str2double(x_2_init_char(i));
    end
    x_2(1:31) = x_2_init;
    for n = 1:1800
        x_1(n+31) = mod(x_1(n+3)+x_1(n),2);
        x_2(n+31) = mod(x_2(n+3)+x_2(n+2)+x_2(n+1)+x_2(n),2);
    end
    for n = 0:119
        c(n+1) = mod(x_1(n+1600+1)+x_2(n+1600+1),2);
    end
    for n = 0:59
        QPSK_stream(n+1) = 1/sqrt(2)*(1-2*c(2*n+1)) + 1j/
sqrt(2)*(1-2*c(2*n+2));
    end
end
```



*Published with MATLAB® R2024a*

---

# Part 4: SSS search & Cell Id Detection

## Table of Contents

Parameters .....	1
SSS .....	1
Channel and Noise .....	1
(a) Plot Correlation for all SNR values .....	2

## Parameters

```
FFT_size = 4096;
CP_length = 288;
SCS = 30e3;
Ts = 1/FFT_size/SCS;
CP_OFDM_length = FFT_size+CP_length;
num_sc = 240;
N_id_1 = 77;
N_id_2 = 2;
```

## SSS

```
% OFDM Modulation
SSS_stream = SSS_BPSK(N_id_1,N_id_2);
% Map symbol to subcarrier
d_SSS = [zeros(56,1);SSS_stream;zeros(FFT_size-183,1)];
% FFT
OFDM_SSS_body = ifft(d_SSS)*sqrt(FFT_size);
% Add CP
CP_OFDM_SSS = [OFDM_SSS_body(end-CP_length+1:end);OFDM_SSS_body];
```

## Channel and Noise

```
h = 1;
signal_after_channel = conv(CP_OFDM_SSS,h);

SNR_values = [-5, 20]; % SNR values in dB
corr_all = zeros(length(SNR_values), 336);

for snr_idx = 1:length(SNR_values)
    SNR = SNR_values(snr_idx);
    N_0 = 10^(-SNR/10) * (norm(signal_after_channel)^2/
length(signal_after_channel));
    noise = sqrt(N_0/2)*(randn(length(signal_after_channel),1) +
1j*randn(length(signal_after_channel),1));
    received_SSS_signal = signal_after_channel + noise;

    corr = zeros(1,336);
```



```

for i = 0:335
    SSS_ref_stream = SSS_BPSK(i,N_id_2);
    d_SSS_ref = [zeros(56,1);SSS_ref_stream;zeros(FFT_size-183,1)];
    OFDM_SSS_ref_body = ifft(d_SSS_ref);
    CP_OFDM_SSS_ref = [OFDM_SSS_ref_body(end-
CP_length+1:end);OFDM_SSS_ref_body];
    corr(i+1) = abs(received_SSS_signal' * CP_OFDM_SSS_ref) ;
end
[~,N_id_1_est_pos] = max(corr);
N_id_1_est = N_id_1_est_pos - 1;
disp(['N_ID1 for SNR = ', num2str(SNR), ' dB: ', num2str(N_id_1_est)]);

% Compute Cell ID using the formula
Cell_ID = 3 * N_id_1_est + N_id_2;
disp(['Cell ID for SNR = ', num2str(SNR), ' dB: ', num2str(Cell_ID)]);

% Store correlation results
corr_all(snr_idx, :) = corr;
end

N_ID1 for SNR = -5 dB: 77
Cell ID for SNR = -5 dB: 233
N_ID1 for SNR = 20 dB: 77
Cell ID for SNR = 20 dB: 233

```

## (a) Plot Correlation for all SNR values

```

figure;
plot(0:335, corr_all(1, :), 'r', 'DisplayName', 'SNR = -5 dB');
hold on;
plot(0:335, corr_all(2, :), 'b', 'DisplayName', 'SNR = 20 dB');
xlabel("Timing Offset (sample)")
ylabel("Correlation")
title('(4a) Correlation for Different SNR Values')
legend show;
grid on;

```

```

function BPSK_stream = SSS_BPSK(N_id_1,N_id_2)
    x_0 = zeros(127,1);
    x_1 = zeros(127,1);
    BPSK_stream = zeros(127,1);
    x_init = [1 0 0 0 0 0 0];
    x_0(1:7) = x_init;
    x_1(1:7) = x_init;

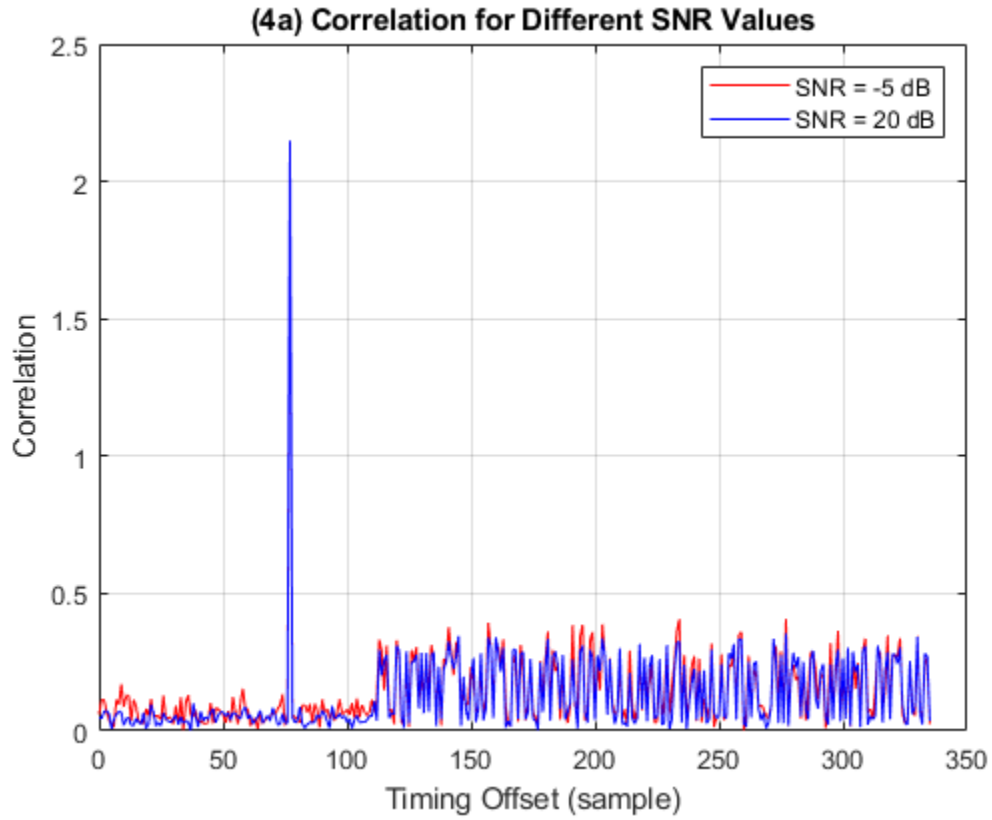
    for i = 1:120
        x_0(i+7) = mod(x_0(i+4)+x_0(i),2);
        x_1(i+7) = mod(x_1(i+1)+x_1(i),2);
    end
    for n = 0:126
        m_0 = mod(n + 15* floor(N_id_1/112) + 5*N_id_2,127);
    end

```

```

m_1 = mod(n + mod(N_id_1,112),127);
BPSK_stream(n+1) = (1-2*x_0(m_0+1))*(1-2*x_1(m_1+1));
end
end

```



*Published with MATLAB® R2024a*

---

# Part 5: Channel Estimation & Equalization

## Table of Contents

Parameters .....	1
PBCH Pilot .....	1
PBCH Data .....	2
Concatenation .....	2
Channel without Noise .....	2
Pilot Subcarriers .....	3
All Subcarriers .....	3
(a) Plot Mean Squared Estimation Error .....	3
(b) Plot SER vs. SNR .....	4

## Parameters

```
FFT_size = 4096;
CP_length = 288;
CP_OFDM_length = FFT_size+CP_length;
num_sc = 240;
SNR = -10:2:10;
num_MC = 1000;

QAM_mod = 4;
c_init = 120897;

h = [1 0.5]';

pilot_subcarriers = 1 + 4 * (0:59) + 1;
data_subcarriers = 1:num_sc;
```

## PBCH Pilot

```
% OFDM Modulation
PBCH_pilot_stream = generate_PBCH_pilot(c_init);
% Map symbol to subcarrier
d_PBCH_pilot = zeros(FFT_size,1);
k = 0:59;
d_PBCH_pilot(1+4*k+1) = PBCH_pilot_stream;
% FFT
OFDM_PBCH_pilot_body = ifft(d_PBCH_pilot)*sqrt(FFT_size);
% Add CP
CP_OFDM_PBCH_pilot = [OFDM_PBCH_pilot_body(end-
CP_length+1:end);OFDM_PBCH_pilot_body];
```

## PBCH Data

```
% OFDM Modulation
PBCH_data_stream = generate_PBCH_data(num_sc, QAM_mod);
% Map symbol to subcarrier
d_PBCH_data = [PBCH_data_stream; zeros(FFT_size-num_sc,1)];
% FFT
OFDM_PBCH_data_body = ifft(d_PBCH_data)*sqrt(FFT_size);
% Add CP
CP_OFDM_PBCH_data = [OFDM_PBCH_data_body(end-
CP_length+1:end); OFDM_PBCH_data_body];
```

## Concatenation

```
CP_OFDM_chain = [CP_OFDM_PBCH_pilot; CP_OFDM_PBCH_data];
```

## Channel without Noise

```
received_signal = conv(CP_OFDM_chain,h);
% OFDM Demodulation
received_CP_OFDM_chain = received_signal(1:CP_OFDM_length*2);
% Remove CP
received_OFDM_pilot_body =
received_CP_OFDM_chain(CP_length+1:CP_OFDM_length);
% FFT
received_pilot = fft(received_OFDM_pilot_body);
% Actual H (actual channel)
H_actual = received_pilot(pilot_subcarriers) ./ PBCH_pilot_stream;

NMSE = zeros(num_MC, length(SNR));
SER = zeros(num_MC, length(SNR));
SER_all = zeros(num_MC, length(SNR));

for SNR_id = 1:length(SNR)
    for MC_id = 1:num_MC

        % Noise
        received_signal_noisy = awgn(received_signal, SNR(SNR_id), 'measured');
        % OFDM Demodulation
        received_CP_OFDM_chain = received_signal_noisy(1:CP_OFDM_length*2);
        % Remove CP
        received_OFDM_pilot_body =
received_CP_OFDM_chain(CP_length+1:CP_OFDM_length);
        received_OFDM_data_body =
received_CP_OFDM_chain(CP_OFDM_length+CP_length+1:end);
        % FFT
        received_pilot = fft(received_OFDM_pilot_body);
        received_data = fft(received_OFDM_data_body);
```

## Pilot Subcarriers

```
% Channel Estimation
ch_est = received_pilot(pilot_subcarriers) ./ PBCH_pilot_stream;

% Normalized Squared Error -->
NMSE(MC_id, SNR_id) = calculate_NMSE(H_actual, ch_est);

% Equalization
equalized_pilot_data = received_data(pilot_subcarriers) ./ ch_est;

% Symbol Error Rate
SER(MC_id, SNR_id) = calculate_SER(equalized_pilot_data,
PBCH_data_stream(pilot_subcarriers), QAM_mod);
```

## All Subcarriers

```
% Channel Estimation
ch_est_full = interp1(pilot_subcarriers, ch_est,
(data_subcarriers)', 'linear', 'extrap');

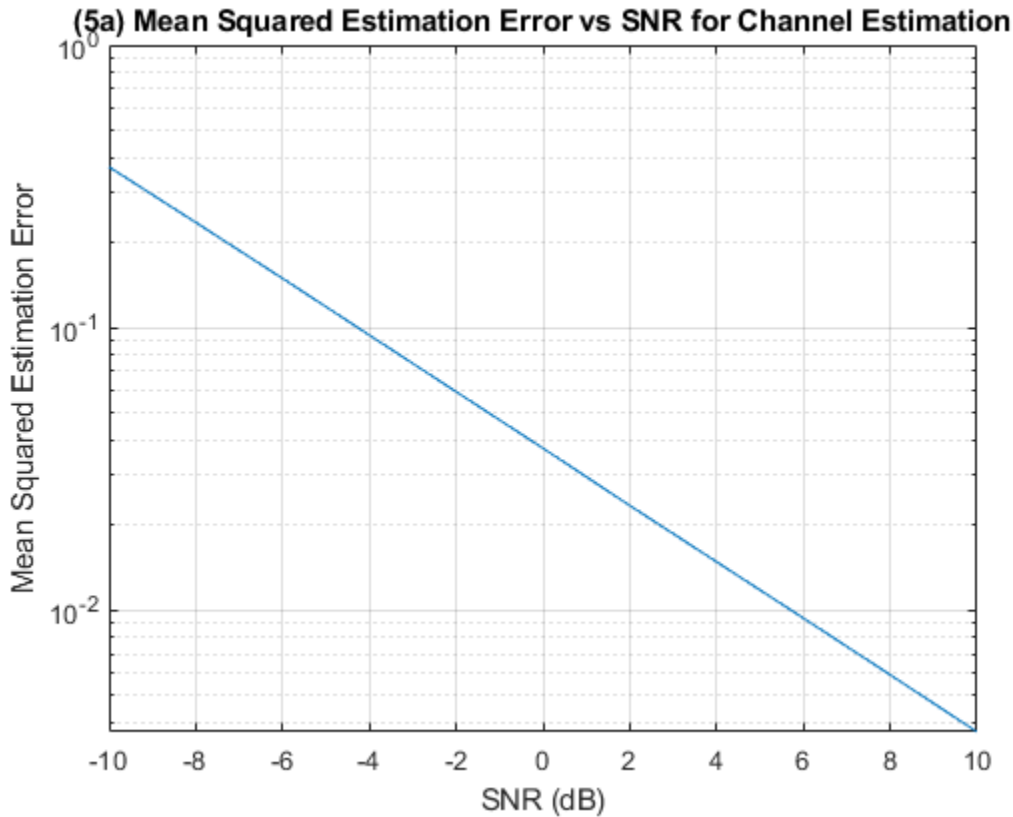
% Equalization
equalized_data = received_data(data_subcarriers) ./ ch_est_full;

% Symbol Error Rate
SER_all(MC_id, SNR_id) = calculate_SER(equalized_data,
PBCH_data_stream, QAM_mod);

end
end
mean_NMSE = mean(NMSE, 1);
mean_SER = mean(SER, 1);
mean_SER_all = mean(SER_all, 1);
```

## (a) Plot Mean Squared Estimation Error

```
figure;
semilogy(SNR, mean_NMSE);
grid on;
xlabel("SNR (dB)");
ylabel("Mean Squared Estimation Error");
title("(5a) Mean Squared Estimation Error vs SNR for Channel Estimation")
```



## (b) Plot SER vs. SNR

```
figure;
semilogy(SNR, mean_SER);
grid on;
xlabel('SNR (dB)');
ylabel('Symbol Error Rate (SER)');
title('(5b) Symbol Error Rate vs. SNR for Pilot Subcarriers');

% (c) Plot SER vs. SNR
figure;
semilogy(SNR, mean_SER_all);
grid on;
xlabel('SNR (dB)');
ylabel('Symbol Error Rate (SER)');
title('(5c) Symbol Error Rate vs. SNR for All Subcarriers');

function QPSK_pilot_stream = generate_PBCH_pilot(c_init)
    c = zeros(120,1);
    QPSK_pilot_stream = zeros(60,1);
    x_1 = zeros(1800,1);
    x_2 = zeros(1800,1);
    x_1_init = [1; zeros(30,1)];
    x_1(1:31) = x_1_init;
    x_2_init = zeros(31,1);
```

```

x_2_init_char = dec2bin(c_init);
for i = 1:length(x_2_init_char)
    x_2_init(length(x_2_init_char)-i+1) = str2double(x_2_init_char(i));
end
x_2(1:31) = x_2_init;
for n = 1:1800
    x_1(n+31) = mod(x_1(n+3)+x_1(n),2);
    x_2(n+31) = mod(x_2(n+3)+x_2(n+2)+x_2(n+1)+x_2(n),2);
end
for n = 0:119
    c(n+1) = mod(x_1(n+1600+1)+x_2(n+1600+1),2);
end
for n = 0:59
    QPSK_pilot_stream(n+1) = 1/sqrt(2)*(1-2*c(2*n+1)) + 1j/
sqrt(2)*(1-2*c(2*n+2));
end
end

function QPSK_data_stream = generate_PBCH_data(num_sc, QAM_mod)
    data_bit_stream = randi([0 1],num_sc*log2(QAM_mod),1);
    QPSK_data_stream =
qammod(data_bit_stream,QAM_mod,InputType='bit',UnitAveragePower=true);
end

function nsme = calculate_NMSE(H_actual, H_estimated)
    % Calculate the squared error between the actual and estimated channel
    responses
    error = H_estimated - H_actual;
    squared_error = abs(error).^2;

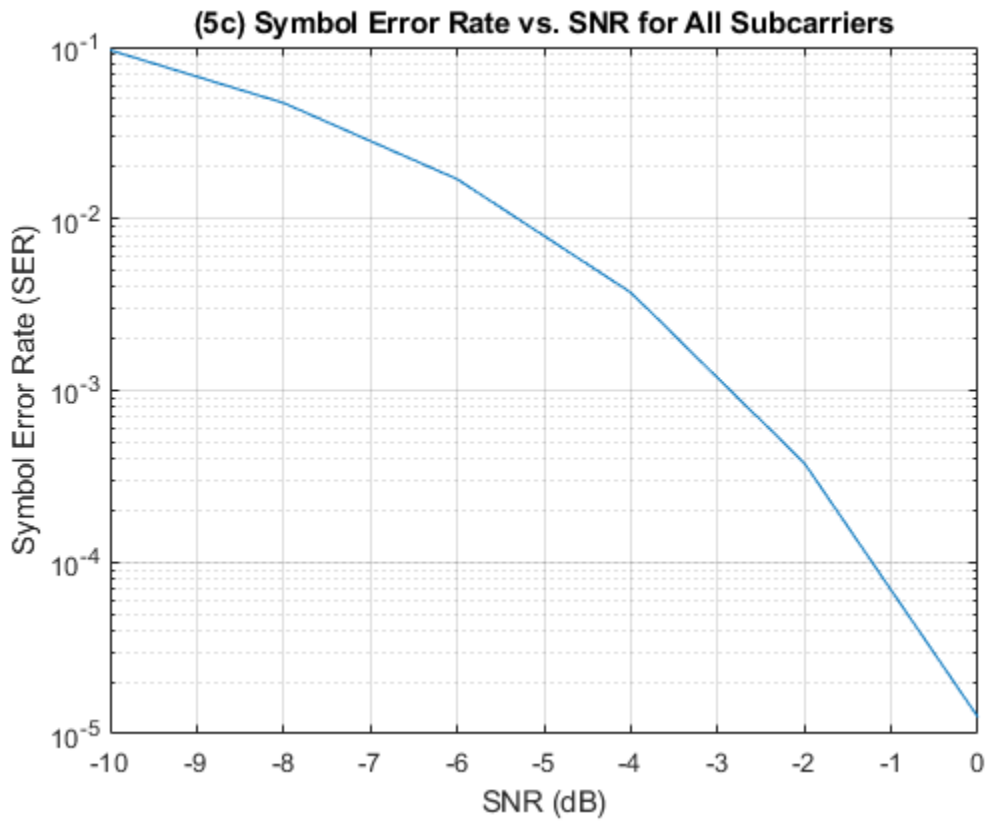
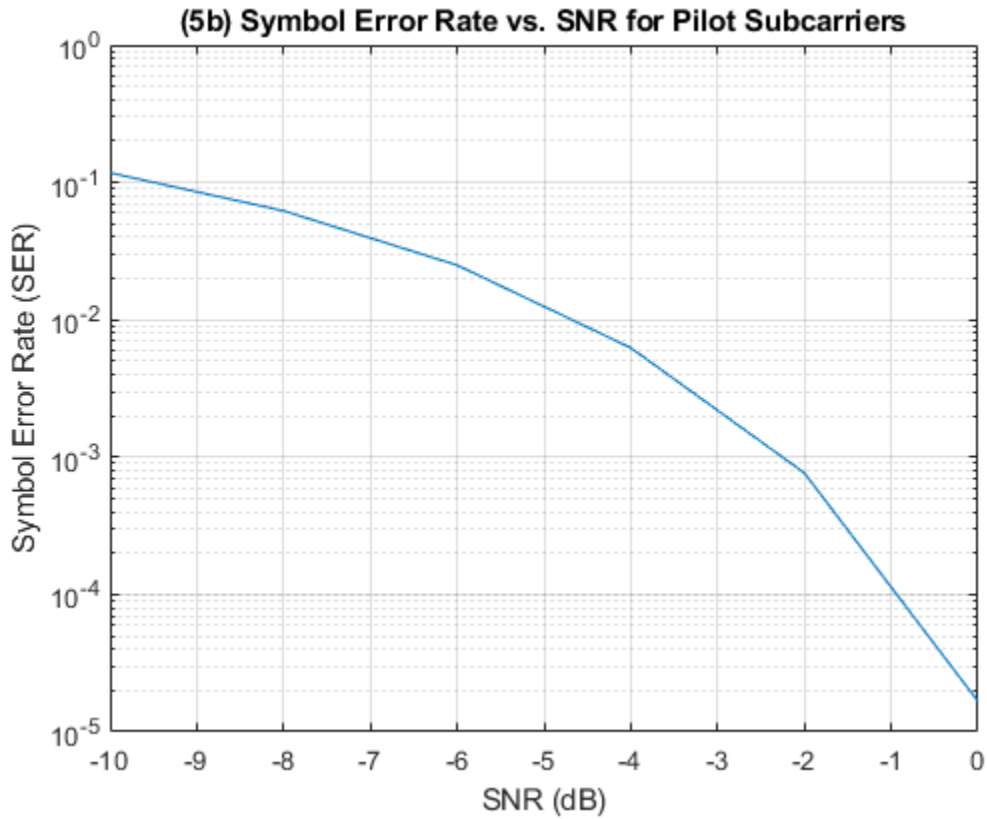
    % Calculate the true power of the actual channel response
    squared_actual = abs(H_actual).^2;

    % Normalize the squared error by the true power --> TODO: Don't need to
    do this??
    nsme = mean(squared_error ./ squared_actual);
end

function ser = calculate_SER(equalized_data, true_data, QAM_mod)
    % QPSK demodulation for both equalized and true data
    estimated_symbols = qamdemod(equalized_data, QAM_mod, 'OutputType',
'bit', 'UnitAveragePower', true);
    true_symbols = qamdemod(true_data, QAM_mod, 'OutputType', 'bit',
'UnitAveragePower', true);

    % Calculate SER
    ser = sum(estimated_symbols ~= true_symbols) / length(true_symbols);
end

```





*Published with MATLAB® R2024a*