Hands-On Lab 2: Study of Digital Modulation Techniques Performance

Objectives:

- Analyze the performance of coherent digital communication systems using BASK, BPSK, BFSK, and QAM modulations.
- Analyze the performance of a non-coherent digital communication system using BDPSK modulation.

Prerequisites:

- Basic knowledge of digital modulation techniques (BASK, BPSK, BFSK, BDPSK, QAM).
- Familiarity with MATLAB.

1. Introduction

Digital modulation techniques are fundamental in modern communication systems. This lab aims to analyze the performance of these techniques in terms of bit error rate (BER) as a function of the signal-to-noise ratio (SNR).

2. Part 1: Coherent System with BASK, BPSK, BFSK, and QAM Modulations

2.1 BASK (Binary Amplitude Shift Keying)

In this part, we will study the performance of a system using BASK modulation.

```
% Coherent Detection for BASK, BPSK, BFSK, and QAM Modulations

clear; close all; clc;

% Parameters
N = 10000;                    % Number of bits
SNR_dB = 0:2:20;              % SNR range in dB
M = 16;                       % QAM Modulation Order (16-QAM)
bitsPerSymbolQAM = log2(M); % Number of bits per symbol for 16-QAM
numSNR = length(SNR_dB);      % Number of SNR points

% Initialize BER arrays for each modulation scheme
BER_BASK = zeros(1, numSNR);
BER_BPSK = zeros(1, numSNR);
BER_BFSK = zeros(1, numSNR);
BER_QAM = zeros(1, numSNR);

% Generate random binary data
data = randi([0 1], 1, N);  % Random bits (0s and 1s)

% Ensure the length of data for QAM is a multiple of bitsPerSymbolQAM
N_qam = floor(N / bitsPerSymbolQAM) * bitsPerSymbolQAM;
data_QAM = data(1:N_qam);  % Adjust data length for QAM

% Reshape data for QAM modulation (group bitsPerSymbolQAM bits per symbol)
data_QAM_reshaped = reshape(data_QAM, bitsPerSymbolQAM, []).';

% Loop over SNR points
for i = 1:numSNR
    % Convert SNR from dB to linear scale
    SNR_linear = 10^(SNR_dB(i)/10);

    %% 1. BASK Modulation and Coherent Detection
    % Modulate: 0 -> 0, 1 -> 1 (BASK)
    baskModulated = data;

    % Add AWGN noise
    noiseVariance = 1/(2*SNR_linear);
    noise = sqrt(noiseVariance) * randn(1, N);
    receivedBASK = baskModulated + noise;

    % Coherent Detection (BASK)
    receivedBits_BASK = receivedBASK > 0.5;  % Simple threshold
    BER_BASK(i) = sum(data ~= receivedBits_BASK) / N;
```

2.2 BPSK (Binary Phase Shift Keying)

Next, we will study the BPSK modulation.

```
%% 2. BPSK Modulation and Coherent Detection
    % Modulate: 0 -> -1, 1 -> 1 (BPSK)
    bpskModulated = 2*data - 1;

    % Add AWGN noise
    noise = sqrt(noiseVariance) * randn(1, N);
    receivedBPSK = bpskModulated + noise;

    % Coherent Detection (BPSK)
    receivedBits_BPSK = receivedBPSK > 0;  % Simple
threshold
    BER_BPSK(i) = sum(data ~= receivedBits_BPSK) / N;
```

2.3 BFSK (Binary Frequency Shift Keying)

Now, we'll explore BFSK modulation.

```
%% 3. BFSK Modulation and Coherent Detection
    % Modulate: 0 -> cos(2*pi*f1*t), 1 -> cos(2*pi*f2*t)
(BFSK)
    f1 = 1; f2 = 2;          % Frequencies for BFSK
    t = (0:N-1)/N;           % Time vector
    bfskModulated = cos(2*pi*f1*t) .* (1 - data) +
cos(2*pi*f2*t) .* data;

    % Add AWGN noise
    noise = sqrt(noiseVariance) * randn(1, N);
    receivedBFSK = bfskModulated + noise;

    % Coherent Detection (BFSK)
    % Demodulate using correlation with each frequency
    corr_f1 = sum(receivedBFSK .* cos(2*pi*f1*t));
    corr_f2 = sum(receivedBFSK .* cos(2*pi*f2*t));
    receivedBits_BFSK = corr_f2 > corr_f1;  % Compare
correlations
    BER_BFSK(i) = sum(data ~= receivedBits_BFSK) / N;
```

2.4 QAM (Quadrature Amplitude Modulation)

Finally, we'll analyze 16-QAM modulation.

```
%% 4. 16-QAM Modulation and Coherent Detection
    % Modulate using QAM (Input must be multiple of bits per
symbol)
    qamModulated = qammod(data_QAM_reshaped, M, 'InputType',
'bit');

    % Add AWGN noise
    noise = (sqrt(noiseVariance/2) * (randn(size(qamModulated)) +
1i*randn(size(qamModulated))));
    receivedQAM = qamModulated + noise;

    % Coherent Detection (QAM)
    receivedBits_QAM = qamdemod(receivedQAM, M, 'OutputType',
'bit');

    % Reshape received bits to match original data format
    receivedBits_QAM_reshaped = reshape(receivedBits_QAM.', 1,
N_qam);

    BER_QAM(i) = sum(data_QAM ~= receivedBits_QAM_reshaped) /
N_qam;

end
```

Display

```
% Plot BER vs SNR for each modulation scheme
figure;
semilogy(SNR_dB, BER_BASK, '-o', 'DisplayName', 'BASK');
hold on;
semilogy(SNR_dB, BER_BPSK, '-s', 'DisplayName', 'BPSK');
semilogy(SNR_dB, BER_BFSK, '-x', 'DisplayName', 'BFSK');
semilogy(SNR_dB, BER_QAM, '-d', 'DisplayName', '16-QAM');
hold off;

title('BER vs SNR for BASK, BPSK, BFSK, and 16-QAM');
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
legend('Location', 'SouthWest');
grid on;
```

3. Part 2: Non-Coherent System with BDPSK Modulation

3.1 BDPSK (Differential Binary Phase Shift Keying)

BDPSK modulation is distinguished by its ability to operate without explicit phase synchronization.

```
% Differential Binary Phase Shift Keying (DBPSK) Modulation and
Demodulation
clear; close all; clc;
% Parameters
N = 10000;                    % Number of bits
SNR_dB = 0:2:20;              % SNR range in dB
numSNR = length(SNR_dB);       % Number of SNR points

% Initialize BER array
BER_DBPSK = zeros(1, numSNR);

% Generate random binary data (0s and 1s)
data = randi([0 1], 1, N);

% Differential encoding
differential_data = zeros(1, N);
differential_data(1) = data(1);  % Assume first bit is the same as
the input bit
for k = 2:N
    differential_data(k) = xor(data(k), differential_data(k-1));
% Differential encoding
end

% Modulate: 0 -> -1, 1 -> 1 (BPSK symbols)
dbpskModulated = 2*differential_data - 1;

% Loop over SNR points
for i = 1:numSNR
    % Convert SNR from dB to linear scale
    SNR_linear = 10^(SNR_dB(i)/10);
% Add AWGN noise
    noiseVariance = 1/(2*SNR_linear);
    noise = sqrt(noiseVariance) * randn(1, N);
    receivedDBPSK = dbpskModulated + noise;
% Coherent detection using differential demodulation
    demodulated_data = zeros(1, N);
    demodulated_data(1) = receivedDBPSK(1) > 0;
% Assume first symbol decoded correctly
      for k = 2:N
        demodulated_data(k) = xor(receivedDBPSK(k-1) > 0,
receivedDBPSK(k) > 0);                                         %
Differential demodulation
    end
% Calculate BER
    BER_DBPSK(i) = sum(data ~= demodulated_data) / N;
end
% Plot BER vs SNR for DBPSK
figure; semilogy(SNR_dB, BER_DBPSK, '-o');
title('BER vs SNR for DBPSK');
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)'); grid on;
```