VIETNAM NATIONAL UNIVERSITY HOCHIMINH CITY
UNIVERSITY OF SCIENCE
FACULTY OF ELECTRONICS AND TELECOMMUNICATION
DEPARTMENT OF TELECOMMUNICATION AND NETWORK

DIGITAL COMMUNICATION

END COURSE PROJECT

Supervisor: Dang Le Khoa, Ph.D
Nguyen Minh Tri, Ph.D
Ngo Thanh Hai, M.Sc

Group Member:

Huynh Thi Ngoc Phuc	21207195
Pham Hoai An	21207120
Tran Thien Phuc	21207077
Do Minh Chuong	21207126

Ho Chi Minh City, Apr 2024

Table of Contents

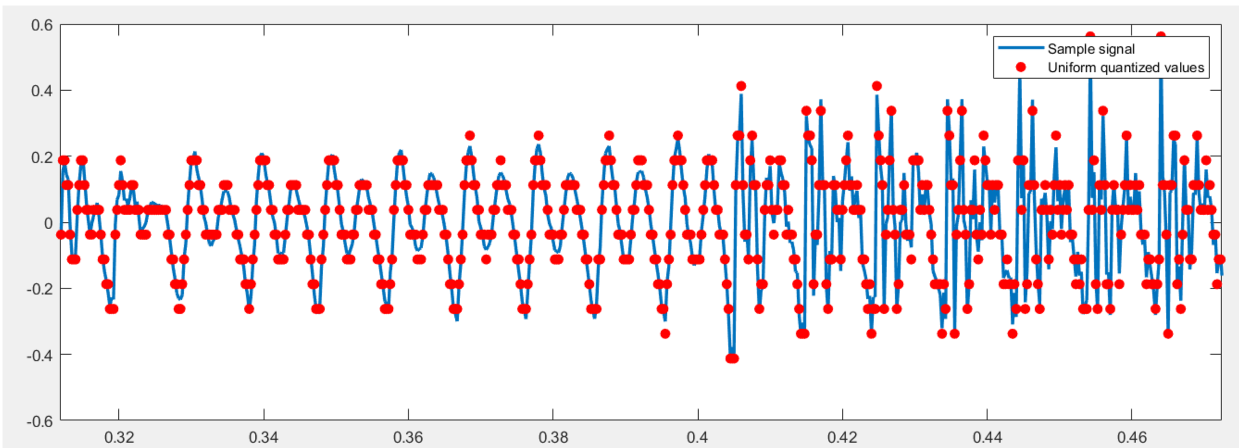
Chapter 2: Formatting and Baseband Modulation

I. Quantize the sample signal ‘mSpeech’

$L = 16$, $q = V\_p/(L-1)$, called $sq2$ signal.
$q = Vpp/(L-1)$ = 2*0.5625 / (16 - 1) = 0.075

Result matlab: 

Plot ‘mSpeech‘ and $sq2$.



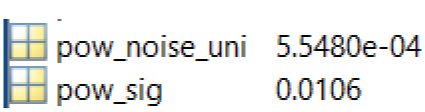
II. Calculate the quantizer

Error variance $(\sigma_{Sq2})^2$ and the ratio of average signal power to average quantization noise power $(S/N)_{Sq2}$ by the numerical method.

pow_noise_uni $= \dfrac{\sum_i^N P_{noise}(i)^2}{N} = 5.5480*10^{-4}\ W$

pow_sig $= \dfrac{\sum_i^N P_{signal}(i)^2}{N} = 0.0106\ W$

SNR_a_uni = pow_sig/pow_noise_uni = *19.0579*



III. Compress the sample signal ‘mSpeech’

called s_{c5}, with μ-law and A-law

- With μ-law

- In North America, a μ-law compression characteristic

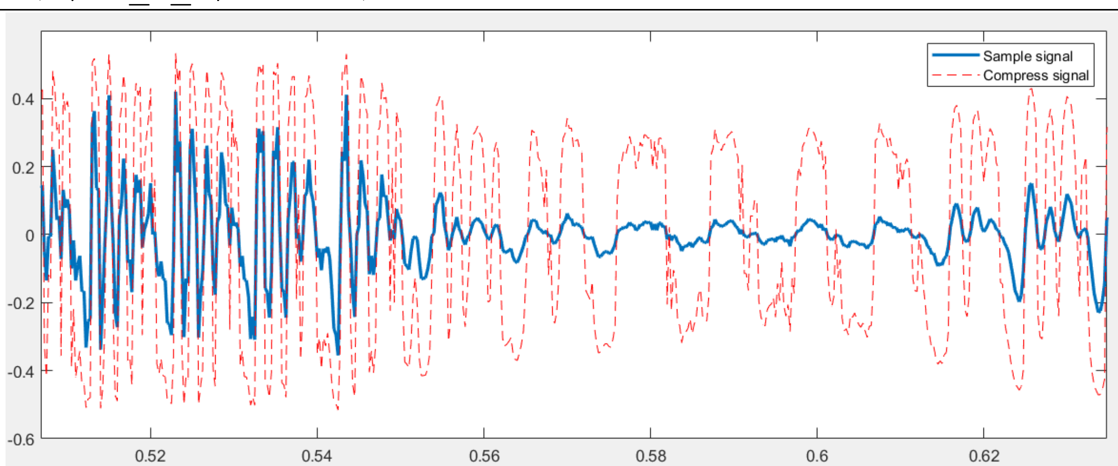
$$y = y_{\max} \frac{\log_e[1 + \mu(|x|)/x_{\max}]}{\log_e(1 + \mu)} \operatorname{sgn} x$$

where

$$\operatorname{sgn} x = \begin{cases} +1 & \text{for } x \geq 0 \\ -1 & \text{for } x < 0 \end{cases}$$

```
function y = ulaw(xmax, ymax, a, mu)
    y = zeros(size(a));
    for i = 1:length(a)
        x = a(i);
        if x >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        y(i)  = ymax * (1/log(1+mu))*log(1+mu*abs(x)/xmax)*sign_x;
    end
end
```

```
s_c_5 = ulaw(x_max, y_max, mSpeech(1:length(t)), mu);
plot(t, s_c_5, 'r--');
```



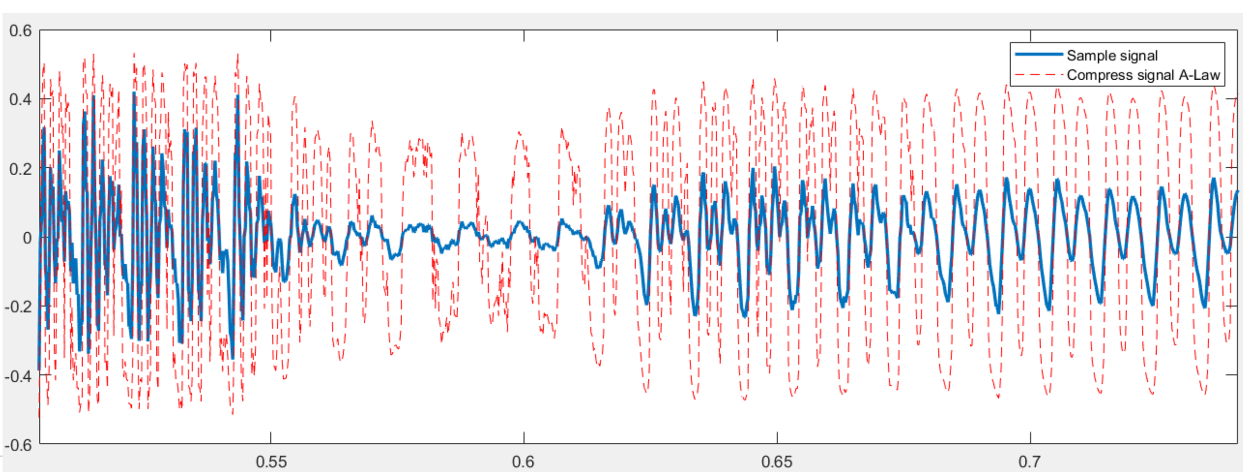
- With A-law

- Another compression characteristic, used mainly in Europe, is the A-law characteristic, defined as

$$y = \begin{cases} y_{\max} \dfrac{A|x|/x_{\max}}{1+\log_e A} \operatorname{sgn} x & 0 < \dfrac{|x|}{x_{\max}} \le \dfrac{1}{A} \\ y_{\max} \dfrac{1+\log_e[A|x|/x_{\max}]}{1+\log_e A} \operatorname{sgn} x & \dfrac{1}{A} < \dfrac{|x|}{x_{\max}} < 1 \end{cases}$$

```
function y = Alaw(xmax, ymax, a, A)
    y = zeros(size(a));
    for i = 1:length(a)
        x = a(i);
        temp = abs(a(i))/xmax;
        if x >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        if (0 < temp && temp <= 1/A)
            y(i) = ymax * sign_x * (A * abs(x) / xmax) / (1
+ log(A));
        elseif (1/A < temp && temp < 1)
            y(i) = ymax * sign_x * (1 + log(A * abs(x) /
xmax)) / (1 + log(A));
        end
    end
end
```

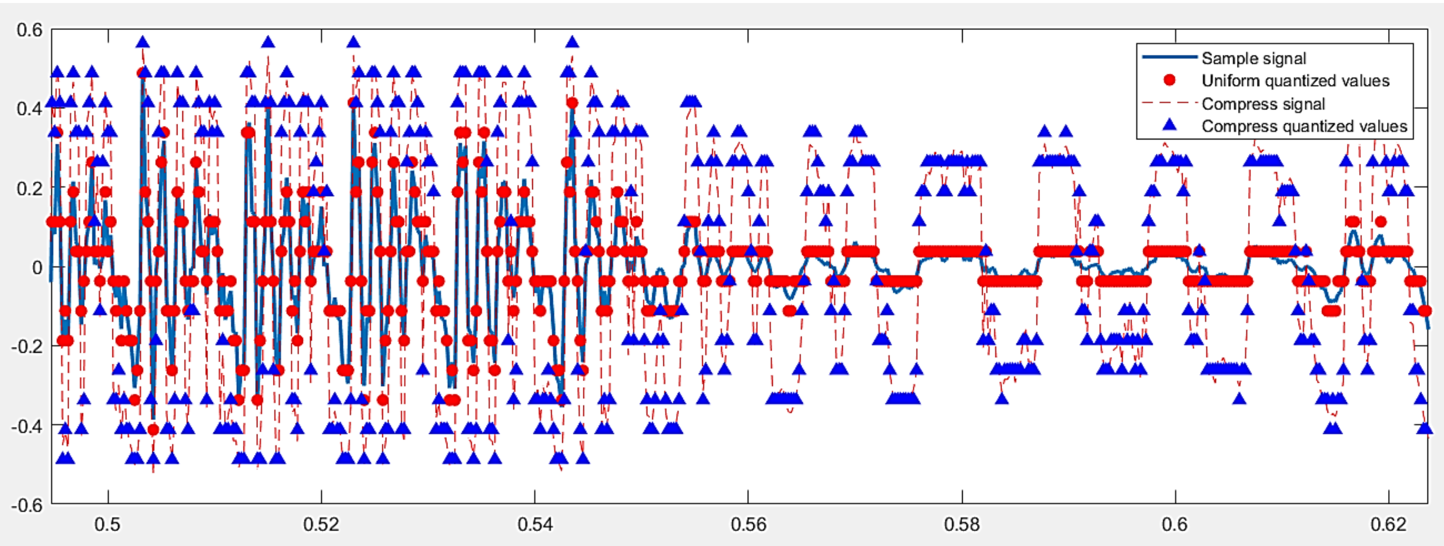
```
s_c_5 = Alaw(x_max, y_max, mSpeech(1:length(t)), A);
plot(t, s_c_5, 'r--');
```



IV. Quantize the compressed signal

s_{c5} with the same parameters as Step 2, called s_{q6}.

```
s_q_6 = quan_uni(s_c_5, q);
plot(t, s_q_6, 'b^', 'MarkerSize', 6, 'MarkerEdgeColor',
'b', 'MarkerFaceColor', 'b');
```



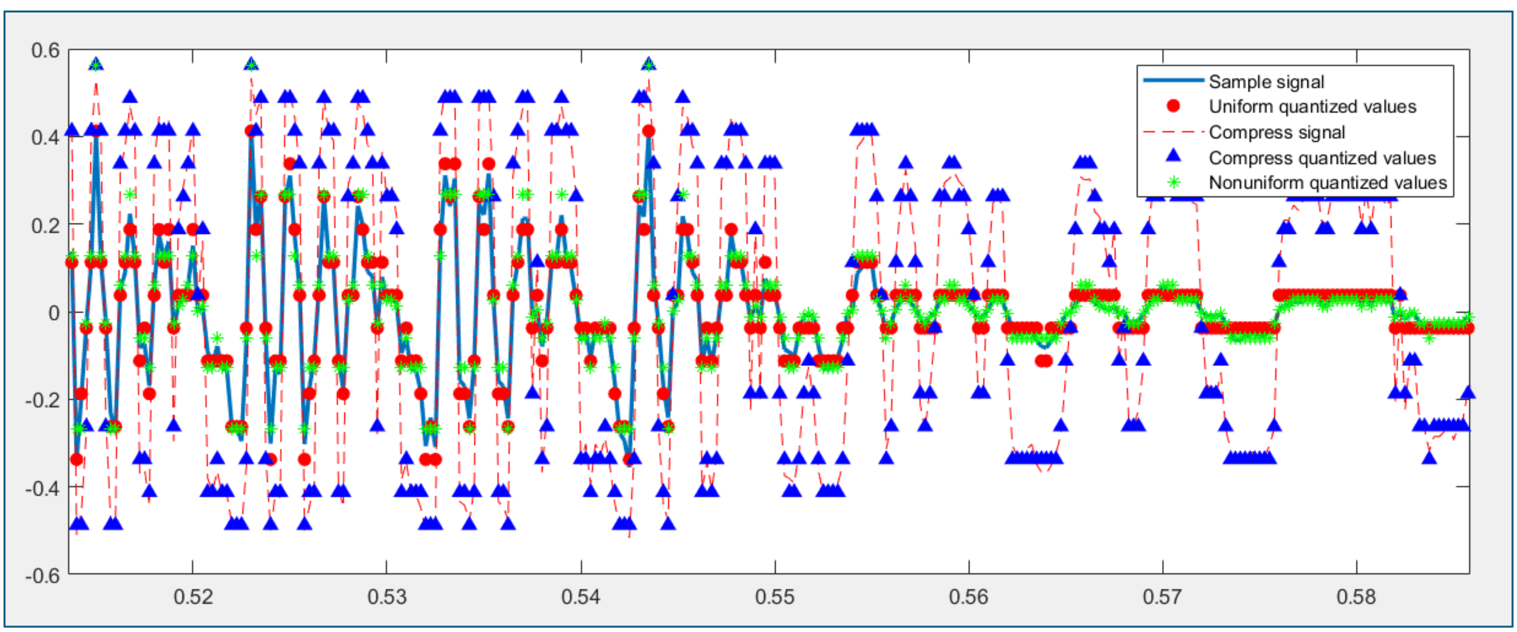
s_{q6} in Step 5, called s_{e7}.

- With μ-law

```
s_e_7 = inv_ulaw(x_max, y_max, s_q_6, mu);

%Function expand mu-law
function x = inv_ulaw(xmax, ymax, y, mu)
    x = zeros(size(y));
    for i = 1:length(y)
        n = y(i);
        if n >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end

        x(i) = (exp(abs(n) * (log(1 + mu)/ ymax) - 1) *
(xmax/mu)) * sign_x;
    end
end
```

- With A-law

```
s_e_7 = inv_Alaw(x_max, y_max, s_q_6, A);

%Function expand A-law
function x = inv_Alaw(xmax, ymax, y, A)
    x = zeros(size(y));
    for i = 1:length(y)
        n = y(i);
        if n >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        x(i) = sign_x * abs(n) * (1 + log(A)) * xmax /
(ymax * A);
        temp = abs(x(i))/xmax;
        if (1/A < temp && temp < 1)
            x(i) = exp(abs(n)*(1+log(A))/ymax-1) * (xmax /
A) * sign_x;
        end
    end
end
```



Calculate $(\sigma Se6)2$ and $(S/N)Se6$.

```
e_com = mSpeech(1:length(t)) - s_e_7;
pow_noise_com = 0;

for i = 1:length(t)
    pow_noise_com = pow_noise_com + e_com(i)^2;
end
pow_noise_com = pow_noise_com / length(t);
SNR_a_com = pow_sig / pow_noise_com;
```

pow_noise_com = $5.2659*10^{-4}$ W
SNR_a_com = pow_sig / pow_noise_com = 20.0789

We have: SNR_a_uni = 19.0579 < SNR_a_com = 20.0789

V. Source Code

```
clear;
% 1. Load speech signal
Fs = 4000;
[mSpeech, Fs] = audioread("MaleSpeech-16-4-mono-
20secs.wav");
% sound(mSpeech,Fs)
% Consider the speech signal in 1.5s
t = 0:1/Fs:1.5;
plot(t, mSpeech(1:length(t)), 'LineWidth', 2);
hold on

% 2. Quantize the sample signal
L = 16; % the number of quantization levels
V_p = 0.5625; % the peak voltage of signal
% Determine the single quantile interval ?-wide
q = 2 * V_p / (L - 1); % Use the exact equation
s_q_2 = quan_uni(mSpeech(1:length(t)), q); % Uniform
quantization
% Plot the sample signal and the quantization signal
plot(t, s_q_2, 'ro', 'MarkerSize', 6, 'MarkerEdgeColor',
'r', 'MarkerFaceColor', 'r');

% 3. Calculate the average quantization noise power,
```

```
% the average power of the sample signal and SNR
e_uni = mSpeech(1:length(t)) - s_q_2; % error between
sample signal and quantized signal

pow_noise_uni = 0;
pow_sig = 0;
    for i = 1:length(t)
        pow_noise_uni = pow_noise_uni + e_uni(i)^2;
        pow_sig = pow_sig + mSpeech(i)^2;
    end
    pow_noise_uni = pow_noise_uni / length(t);
    pow_sig = pow_sig / length(t);
    SNR = pow_sig / pow_noise_uni;
% --------compression-------------

% 5. Compress the sample signal 'mSpeech'
mu = 255; %mu-Law
% A = 87.6; %use the standard value A-Law
y_max = V_p;
x_max = V_p;
% Replace the compress equation for u-law and A-law
% with x is the 'mSpeech' signal
s_c_5 = ulaw(x_max, y_max, mSpeech(1:length(t)), mu); %mu-
Law
% s_c_5 = Alaw(x_max, y_max, mSpeech(1:length(t)), A); %A-
Law
% Plot the compress signal;
plot(t, s_c_5, 'r--');

% 6. Quantize the compress signal and plot the quantized
signal
s_q_6 = quan_uni(s_c_5, q);
plot(t, s_q_6, 'b^', 'MarkerSize', 6, 'MarkerEdgeColor',
'b', 'MarkerFaceColor', 'b');

% 7. Expand the quantized signal
s_e_7 = inv_ulaw(x_max, y_max, s_q_6, mu); %mu-Law
% s_e_7 = inv_Alaw(x_max, y_max, s_q_6, A); %A-Law

plot(t, s_e_7, 'g*', 'MarkerSize', 6, 'MarkerEdgeColor',
'g', 'MarkerFaceColor', 'g');

legend('Sample signal', 'Uniform quantized values',
'Compress signal', ...
```

```
    'Compress quantized values', 'Nonuniform quantized
values');

% 9. Calculate the average quantization noise power,
% the average power of the analog signal and SNR
e_com = mSpeech(1:length(t)) - s_e_7;
pow_noise_com = 0;

for i = 1:length(t)
    pow_noise_com = pow_noise_com + e_com(i)^2;
end
pow_noise_com = pow_noise_com / length(t);
SNR_a_com = pow_sig / pow_noise_com;

%%Function
function quan_sig = quan_uni(sig, q)
    for i = 1:length(sig)
        quan_sig(i) = quantize(sig(i), q);
        d = sig(i) - quan_sig(i);
        if d == 0
            quan_sig(i) = quan_sig(i) + q/2;
        elseif (d > 0) && (abs(d) < q/2)
            quan_sig(i) = quan_sig(i) + q/2;
        elseif (d > 0) && (abs(d) >= q/2)
            quan_sig(i) = quan_sig(i) - q/2;
        elseif (d < 0) && (abs(d) < q/2)
            quan_sig(i) = quan_sig(i) - q/2;
        elseif (d < 0) && (abs(d) >= q/2)
            quan_sig(i) = quan_sig(i) + q/2;
        end
    end
end

%Function compand mu-law
function y = ulaw(xmax, ymax, a, mu)
    y = zeros(size(a));
    for i = 1:length(a)
        x = a(i);
        if x >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
```

```
        y(i) = ymax *
(1/log(1+mu))*log(1+mu*abs(x)/xmax)*sign_x;
    end
end

%Function expand mu-law
function x = inv_ulaw(xmax, ymax, y, mu)
    x = zeros(size(y));
    for i = 1:length(y)
        n = y(i);
        if n >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        x(i) = (exp(abs(n) * (log(1 + mu)/ ymax) - 1) *
(xmax/mu)) * sign_x;
    end
end

%Function compand A-law
function y = Alaw(xmax, ymax, a, A)
    y = zeros(size(a));
    for i = 1:length(a)
        x = a(i);
        temp = abs(a(i))/xmax;
        if x >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        if (0 < temp && temp <= 1/A)
            y(i) = ymax * sign_x * (A * abs(x) / xmax) / (1
+ log(A));
        elseif (1/A < temp && temp < 1)
            y(i) = ymax * sign_x * (1 + log(A * abs(x) /
xmax)) / (1 + log(A));
        end
    end
end

%Function expand A-law
function x = inv_Alaw(xmax, ymax, y, A)
    x = zeros(size(y));
```

```
    for i = 1:length(y)
        n = y(i);
        if n >= 0
            sign_x = 1;
        else
            sign_x = -1;
        end
        x(i) = sign_x * abs(n) * (1 + log(A)) * xmax /
(ymax * A);
        temp = abs(x(i))/xmax;
        if (1/A < temp && temp < 1)
            x(i) = exp(abs(n)*(1+log(A))/(ymax)-1) * (xmax
/ A) * sign_x;
        end
    end
end
```

Chapter 3: Formatting and Baseband Modulation

I. What is Conditional Probability?

1. Definition

Conditional probability is known as the probability that an event or result will appear based on the occurrence of a prior event or result. The likelihood of the prior event is multiplied by the current probability of the subsequent, or conditional, event to determine the conditional probability.

Conditional probability can be distinct with unconditional probability. Unconditional probability refers to the likelihood that a situation will occur, regardless of whether previous events or other conditions happened.

2. How Does It Works?

Conditional probabilities are dependent on an earlier outcome or occurrence happening. A conditional probability would examine the connections between these occurrences.

When one event's occurrence has no bearing on the possibility of the other event happening, two events are said to be independent. However, two events are said to be dependent if one event's occurrence or non-occurrence actually affects the probability that the other event will occur.

The probability of an event B does not depend on what occurs with event A if the events are independent. Therefore, an event that depends on another has a conditional probability.

The "probability of A given B," referred to as P(A|B), is a common representation of conditional probability.

3. Formula of Conditional Probability

$$P(B|A) = \frac{P(A \cap B)}{P(B)}$$

Where: P is the probability

A is the event A

B is the event B

II. Performance with binary signaling.

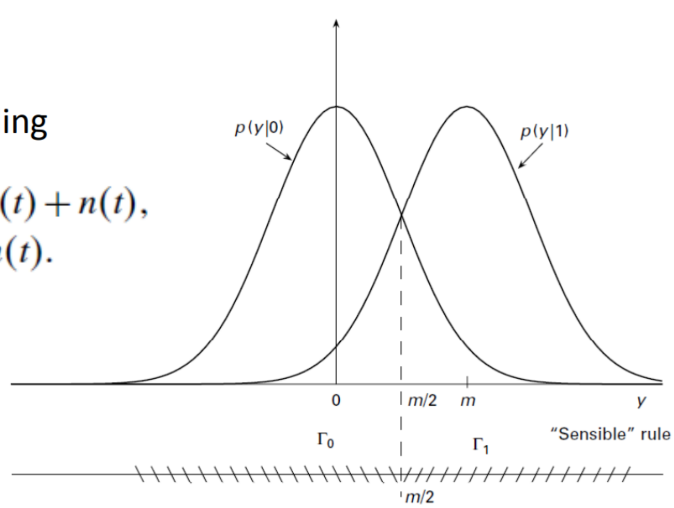
The basic building block for performance analysis is binary signaling. Specifically, consider on–off signaling with.

$$\left.\begin{array}{l} H_1 : y(t) = s(t) + n(t), \\ H_0 : y(t) = n(t). \\ \\ \delta_{\text{ML}}(y) = \arg \max\limits_{1 \le i \le M} \langle y, s_i \rangle - \dfrac{||s_i||^2}{2}. \end{array}\right\} \quad \langle y, s \rangle \underset{H_0}{\overset{H_1}{\gtrless}} \frac{||s||^2}{2}.$$

Setting $Z = \langle y, s \rangle$ which is the decision statistic. Conditioned on either hypothesis, Z is a Gaussian random variable. We wish to compute the conditional error probabilities.

on–off signaling

$$H_1 : y(t) = s(t) + n(t),$$
$$H_0 : y(t) = n(t).$$



The ML rule is given by

$$\langle y, s\rangle \underset{H_0}{\overset{H_1}{\gtrless}} \frac{||s||^2}{2}. \xrightarrow{Z = \langle y, s\rangle} Z \underset{H_0}{\overset{H_1}{\gtrless}} \frac{m}{2}$$

and its performance is given by $P_{e,ML} = Q(\frac{m}{2v})$

Energy per bit, Eb

A design is more power efficient if it gives the same performance with a smaller Eb, if we fix the noise strength. Assuming that 0 and 1 are equally likely to be sent,

$$E_b = \frac{1}{2}(\|s_0\|^2 + \|s_1\|^2)$$

Performance scaling with signal and noise strengths

If we scale up both s1 and s0 by factor A, Eb scales up by a factor A 2, while the distance distance d scales up by a factor A. We therefore therefore define the scale - invariant parameter

$$\eta_P = \frac{d^2}{E_b}.$$

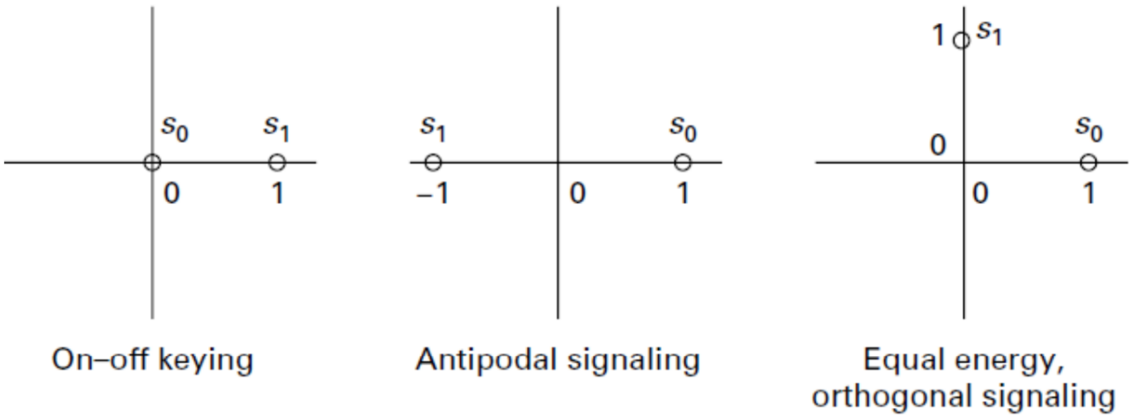
$$\left.\begin{array}{r} d = \sqrt{\eta_P E_b} \\ \sigma = \sqrt{N_0/2} \\ P_{e,ML} = P_{e|1} = P_{e|0} = Q\left(\frac{||s_1 - s_0||}{2\sigma}\right) = Q\left(\frac{d}{2\sigma}\right) \end{array}\right\} P_{e,ML} = Q\left(\sqrt{\frac{\eta_P E_b}{2N_0}}\right) = Q\left(\sqrt{\frac{d^2}{E_b}}\sqrt{\frac{E_b}{2N_0}}\right).$$

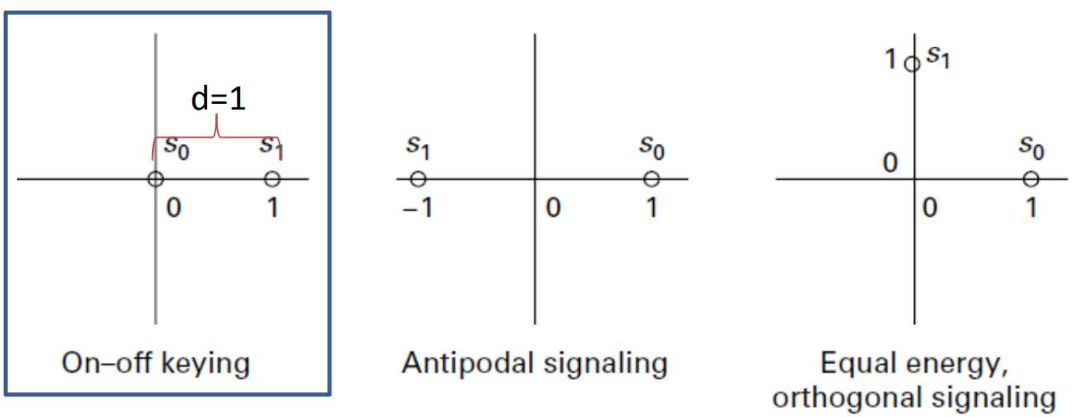
Performance depends on signal to noise ratio

We observe from in that the performance depends on the SNR Eb/N0, rather than separately on the signal and noise strengths. Concept of power efficiency For fixed Eb/N0, the performance is better for a signaling scheme that has a higher value of $^{\eta}{}_P * {}^{\eta}{}_P = \frac{d^2}{E_b}$ is called power efficiency.

Let us now compute the performance of some common binary signaling schemes schemes in terms of Eb/No

$$P_{e,ML} = Q\left(\sqrt{\frac{\eta_P E_b}{2N_0}}\right) = Q\left(\sqrt{\frac{d^2}{E_b}}\sqrt{\frac{E_b}{2N_0}}\right).$$

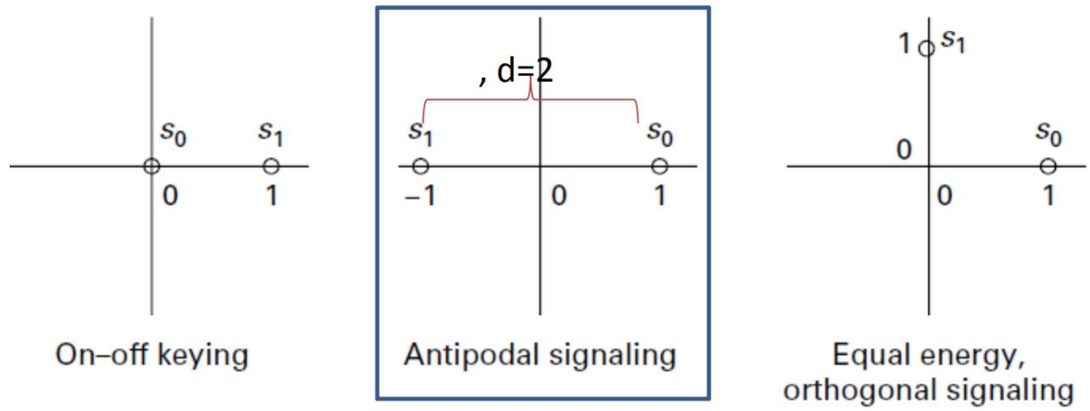




On–off keying (OOK) $s_1(t) = s(t)\ and\ s_0(t) = 0$

The signal space is one-dimensional. For the scaling in the figure, we have d=1 and $E_b = \frac{1}{2}(1^2 + 0^2) = \frac{1}{2}\ so\ that\ \eta_p = \frac{d^2}{E_b} = 2\ .By\ using$

$$P_{e,ML} = Q\left(\sqrt{\frac{\eta_P E_b}{2N_0}}\right) = Q\left(\sqrt{\frac{d^2}{E_b}}\sqrt{\frac{E_b}{2N_0}}\right). \Longrightarrow \quad P_{e,ML} = Q(\sqrt{E_b/N_0}).$$

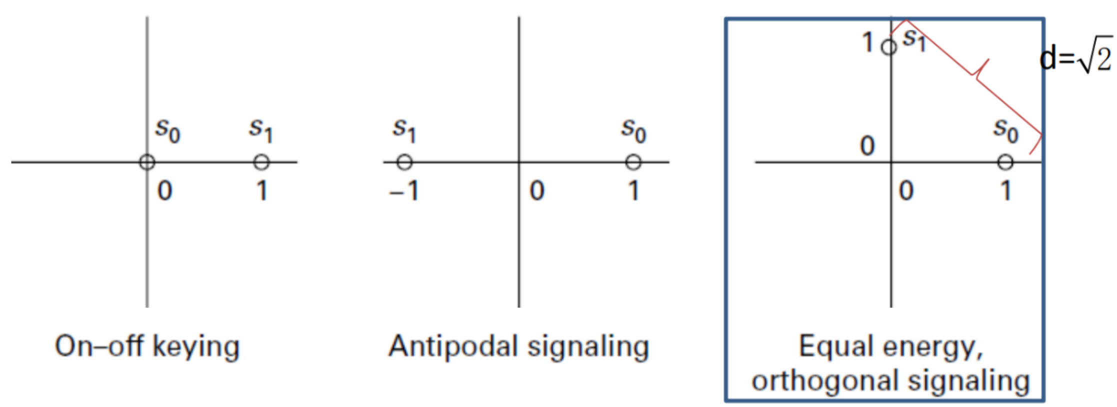


Antipodal signaling $s_1(t) = -s_0(t)$

A one-dimensional signal space representation. One possible realization of antipodal signaling is BPSK.

For the scaling chosen, d=2 and $E_b = \frac{1}{2}(1^2 + (-1)^2) = 1\ so\ that\ \eta_P = \frac{d^2}{E_b} = 4$. By using.

$$P_{e,ML} = Q\left(\sqrt{\frac{\eta_P E_b}{2N_0}}\right) = Q\left(\sqrt{\frac{d^2}{E_b}}\sqrt{\frac{E_b}{2N_0}}\right). \Longrightarrow P_{e,ML} = Q(\sqrt{2E_b/N_0})$$



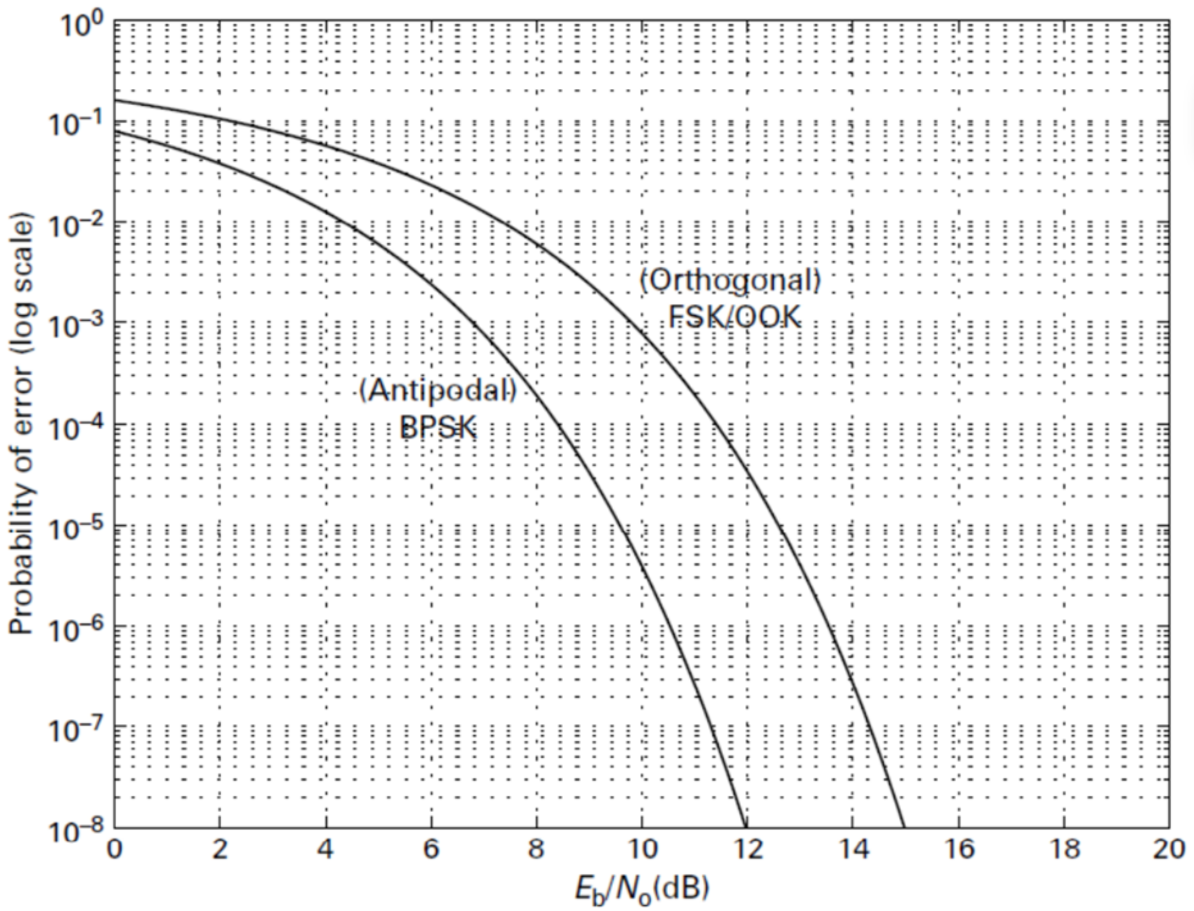
Equal-energy orthogonal signaling

Here the signals are orthogonal and $||s_0||^2 = ||s_1||^2$

This is a two-dimensional signal space. An example for this kind of signaling is frequency shift keying (FSK)

$$d = \sqrt{2}\ and\ E_b = 1\ \eta_P = \frac{d^2}{E_b} = 2$$

$$P_{e,ML} = Q\left(\sqrt{\frac{E_b}{N_O}}\right).$$



III. Marginal and Joint Probability

1. Marginal probability

The likelihood that an event will occur (p(A)) by itself. It could be related to an unconditional probability. It is not dependent on a separate event.

2. Joint probability

Joint probability is that of event A and event B occurring. It is a possibility that multiple events will occur at the same time. The chance that A and B will intersect can be stated as $p(A \cap B)$.

IV. Bayes Theorem

1. Definition

Bayes' theorem is a mathematical formula for determining conditional probability called after the 18th-century British mathematician Thomas Bayes.

The theorem offers a method to update probabilities for existing theories or predictions in light of fresh or additional data. The Bayes theorem can be used in finance to determine the risk of lending money to prospective borrowers.

The concept of Bayesian statistics is based on the Bayes' theorem, also known as Bayes' Rule or Bayes' Law. This set of probability principles helps one to revise their predictions of events happening in light of fresh information, leading to better and more dynamic estimates.

2. Formula of Bayes Theorem

$$P(B|A) = \frac{P(B|A) * P(A)}{P(B)}$$

Where: P is the probability

A is the event A

B is the event B.

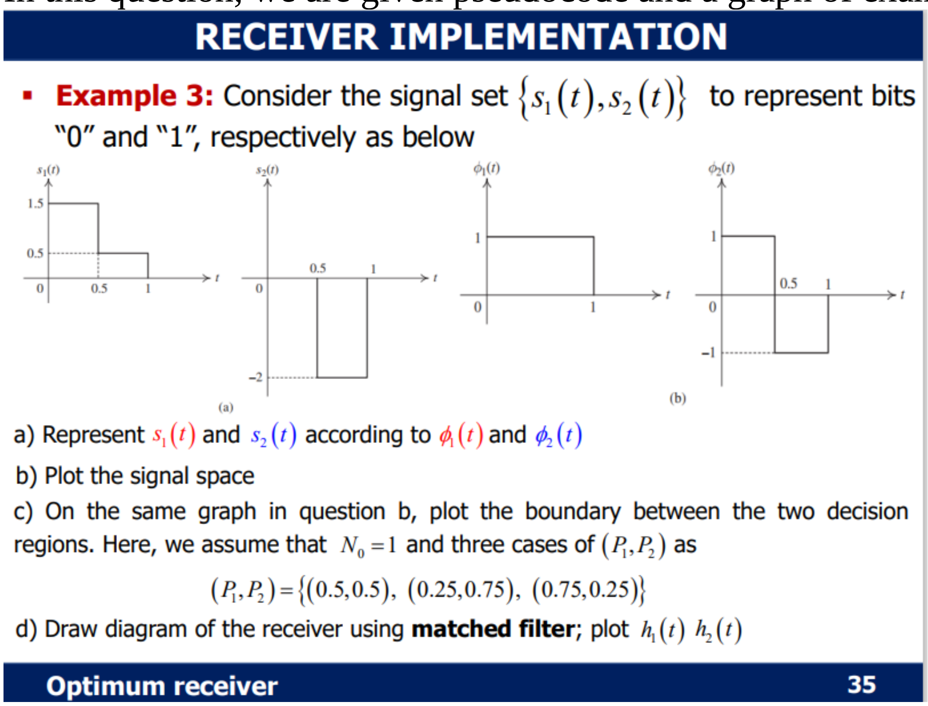
V. Conclusion

Conditional probability is a measure of the probability of an event occurring, given that another event (by assumption, presumption, assertion or evidence) has already occurred.

VI. Project Chapter 3

Question 1: Using the hyper thesis in Example 3, complete the following code to present 10 bits of the signal, which is transmitted with $P_1 = P_2 = 0.5$.

In this question, we are given pseudocode and a graph of example 3.



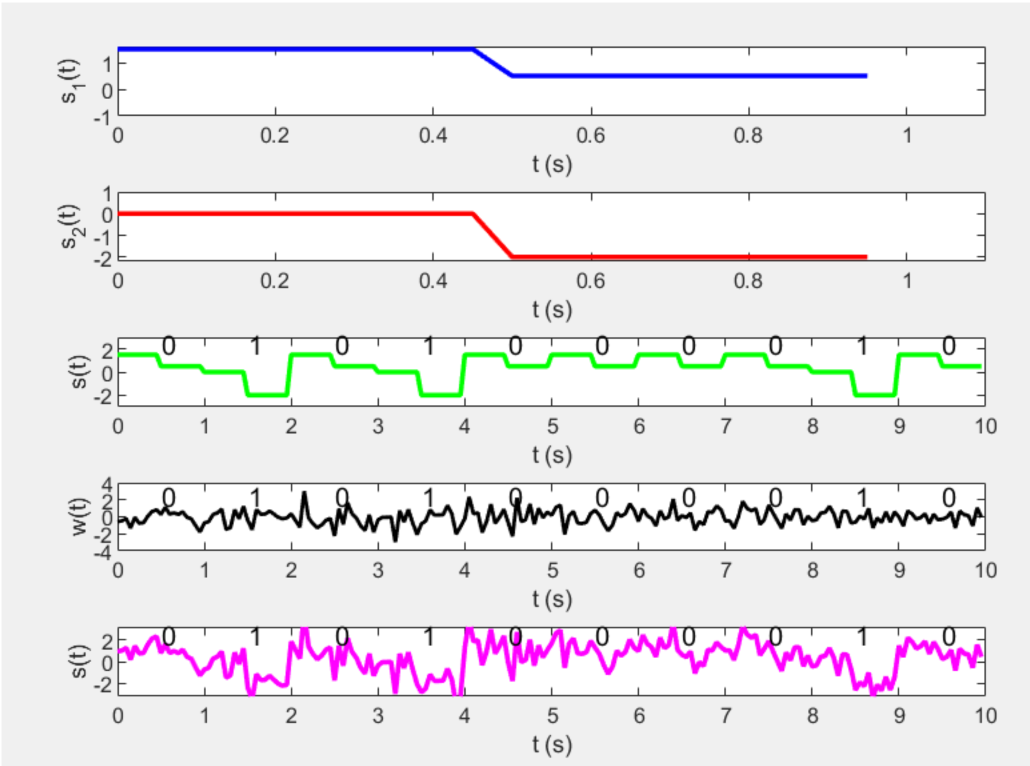
For the pseudocode, complete the sections marked with "???" and finish the code according to the correct title.

Source Code:

```
clc; clear; close all;
% ================ Represent s1(t) and s2(t)
ts = 0.05; % The sample time
t1 = 0: ts: 0.5 - 0.05;
t2 = 0.5: ts: 1 - 0.05;
t_1bit = [t1 t2]; % Time of 1 bit
L = length(t_1bit); % The number of samples of 1 bit
s1_t1 = 1.5;
s1_t2 = 0.5;
s1 = [ones(1, length(t1)) * s1_t1 ones(1, length(t2))*s1_t2]; % s1(t)
s2_t1 = 0;
```

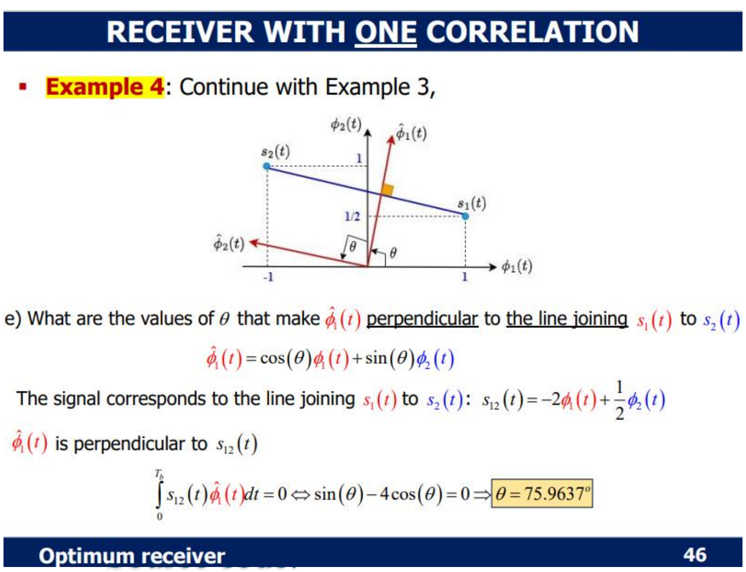
```
s2_t2 = -2;
s2 = [ones(1, length(t1)) * s2_t1 ones(1, length(t2))*s2_t2]; % s2(t)
% ================ The transmitted signal
Ntry = 10^1; % The total transmitted bits
Bit = randi([0 1], 1, Ntry);% Transmission with P1 = P2 = 0.5

s = []; % The transmitted signal s(t)
t = []; % The time of s(t)
for i = 1:Ntry
    if Bit(i) == 0
        s = [s s1];
    else
        s = [s s2];
    end
    t_ibit = (i - 1) * L * ts + t_1bit; % Thời gian của i-bit
    t = [t t_ibit];
end

% ================= The AWGN channel
N0_2 = 0.05; % The noise power spectrum density (W/Hz) N0/2
B = 1/ts; % Bandwidth of signals
Power_noise = N0_2 * B; % The power of noise

% Generate white noise
% w = sqrt(Power_noise/2) * (randn(size(s)) + 1j*randn(size(s)));
w = sqrt(Power_noise) * randn(1, length(s));
r = s + w; % Add noise to the transmitted signal
figure(1)
subplot(5,1,1)
plot(t_1bit,s1,'b-','linewidth',1.8); hold on;
xlabel('t (s)'); ylabel('s_1(t)');
axis([0 1.1 -1 1.6])
subplot(5,1,2)
plot(t_1bit,s2,'r-','linewidth',1.8);
xlabel('t (s)'); ylabel('s_2(t)')
axis([0 1.1 -2.2 1])
x_note = 0.5 :1 :Ntry - 0.5;
y_note = 2.4 *ones(1,Ntry);
Text = string(Bit);
subplot(5,1,3)
plot(t,s,'g-','linewidth',1.8);
text(x_note, y_note, Text);
xlabel('t (s)'); ylabel('s(t)')
axis([0 Ntry -3 3])
subplot(5,1,4)
plot(t,w,'k-','linewidth',1.4);
text(x_note, y_note, Text);
xlabel('t (s)'); ylabel('w(t)')
axis([0 Ntry -4 4])
subplot(5,1,5)
plot(t,r,'m-','linewidth',1.8);
text(x_note, y_note, Text);
xlabel('t (s)'); ylabel('s(t)')
axis([0 Ntry -3.2 3.2])
```

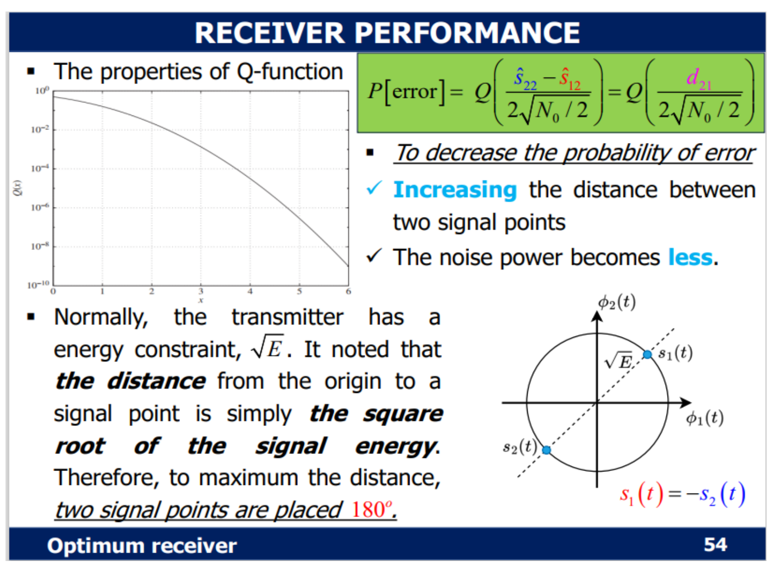
Conclusion :



Question 2: Based on the receiver implementation in Example 4, complete the following code to evaluate the system performance via the bit error probability.

In this question, we are given pseudocode and a graph of example 4.



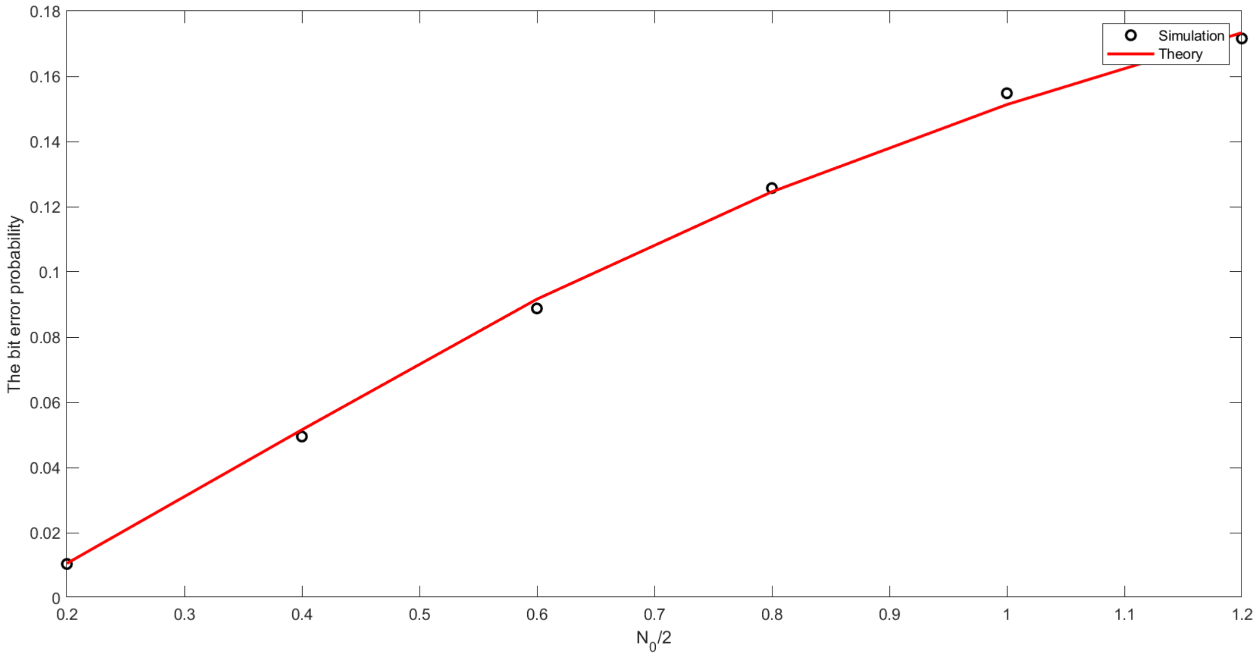



```
clc; clear; close all;
```

```
% ================ Represent s1(t) and s2(t)
ts = 0.1; % The sample time
t1 = 0: ts: 0.5 - 0.05;
t2 = 0.5: ts: 1 - 0.05;
t_1bit = [t1 t2]; % Time of 1 bit
L = length(t_1bit); % The number of samples of 1 bit
s1_t1 = 1.5;
s1_t2 = 0.5;
s1 = [ones(1, length(t1)) * s1_t1 ones(1, length(t2))*s1_t2]; % s1(t)
s2_t1 = 0;
s2_t2 = -2;
s2 = [ones(1, length(t1)) * s2_t1 ones(1, length(t2))*s2_t2]; % s2(t)

% ================ The transmitted signal
Ntry = 10^4; % The total transmitted bits
N0_2 = 0.2:0.2:1.2; % The noise power spectrum desity (W/Hz) N0/2
P_error_simul = zeros(1,length(N0_2));
P_error_theo = zeros(1,length(N0_2));
for j = 1:length(N0_2)
     Bit = randi([0 1], 1, Ntry, 'double'); % Transmission with P1 = P2;
     s = []; % The transmitted signal s(t)
     t = []; % The time of s(t)
    for i = 1:Ntry
        if Bit(i) == 0
            s = [s s1];
        else
            s = [s s2];
        end
        t_ibit = (i - 1) * L * ts + t_1bit; % Thời gian của i-bit
        t = [t t_ibit];
    end
     % ================= The AWGN channel
     B = 1/ts; % Bandwidth of signals
     Power_noise = N0_2(j) * B; % The power of noise
     w = sqrt(Power_noise) * randn(1, length(s));
     % ================= The received signal
     r = s + w; % Add noise to the transmitted signal
     % ================= The recovered signal
     h = [ones(1, length(t1)) * (-5/sqrt(17)) ones(1, length(t2))*(-3/sqrt(17))];
     %h_t2 = [ones(1, length(t1)) * (-3*sqrt(5)/5) ones(1, length(t2))*(sqrt(5)/5)];
     %h = [h_t1 h_t2]; % The impulse response of the matched filter
     T = 3/(4 * sqrt(17)); % The decision threshold
     Bit_rec = zeros(1,Ntry);
     for i = 1:Ntry
         Frame = r((i-1)*L+1 : i*L); % Construct 1 Frame with L samples
         y = conv(Frame, h) * ts; % The signals pass through the matched filter
         r2_mu = y(L);
         % --------- Comparator for decision
         if r2_mu >= T
            Bit_rec(i) = 1;
         else
            Bit_rec(i) = 0;
         end
     end
     Bit_rec;
```

```
    % ================== The bit error probability
    % ------------- Simulation
    [Num, rate] = biterr(Bit, Bit_rec);
    P_error_simul(j) = rate;
    % ------------- Theory
    s12_mu = (-7*sqrt(17)/34);
    s22_mu = (5*sqrt(17)/17);
    P_error_theo(j) = qfunc((s22_mu - s12_mu)/(2 * sqrt(N0_2(j))));
end
figure(1)
plot(N0_2,P_error_simul,'ko','linewidth',1.6,'markersize',6);
hold on;
plot(N0_2,P_error_theo,'r-','linewidth',1.8,'markersize',6);
xlabel('N_0/2'); ylabel('The bit error probability');
legend('Simulation','Theory');
```

Conclusion



Chapter 4: Basic Digital Passband Modulation

I. QPSK Modulation.

QPSK (Quadrature Phase Shift Keying) is quadrature phase modulation. In this technique, the data to be transmitted will be transmitted in sets of 2 bits, each set of 2 bits is called a symbol. Each phase position is a symbol

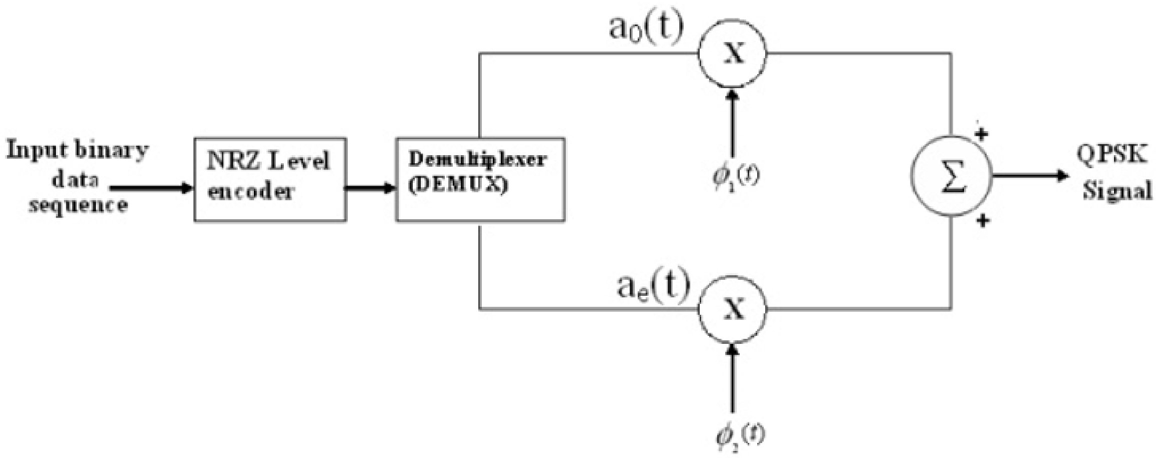
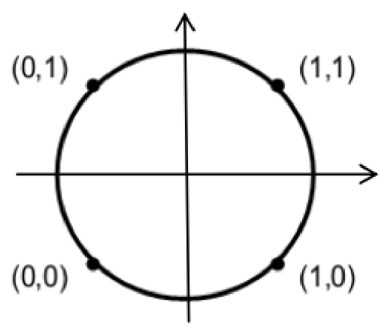


Fig 1. Block diagram of QPSK modulation

Phase diagram QPSK:

Symbol	Phase
11	45^0
01	135^0
00	225^0
10	315^0



II. QPSK Demodulation.

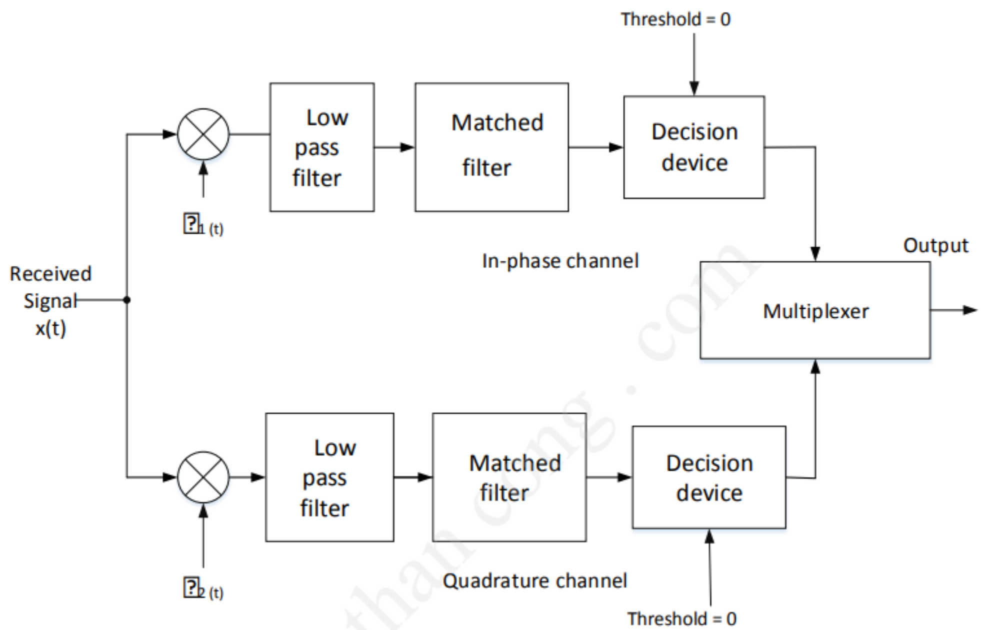


Fig 2. Block diagram of QPSK demodulation

operating principle:

Stage 1: convert the signal at the bandpass band r(t) to the signal at the lowband band (baseband) by multiplying it with the corresponding carrier signal for the purpose of eliminating the carrier component.

Multiply the signal by $\sin(2\pi f_c t)$ we obtain signal I, multiply the signal by $\cos(2\pi f_c t)$ we receive the Q signal.

Baseband signal:

$$r(t) = A_c\cos(2\pi f_c t + \varphi) \text{ với} \varphi = \pi(n-1) \text{ with } n = (0, 1 \text{ the input bits})$$

We have:

$$v(t) = r(t)A_c\cos(2\pi f_c t) = A_c\cos(2\pi f_c t).A_c\cos(2\pi f_c t)$$

Then pass v(t) through a low-pass filter (LPF). The effect of the low-pass filter helps eliminate high-frequency components of the carrier wave, retaining low-frequency components (baseband signal)

$$z = \int_0^{T_b} v(t)dt = \int_0^{T_b} A_c^2\,[\cos(2w_c t + \varphi) + cos\varphi]$$

With Tb is time to transmit 1-bit

Stage 2: is the decisive stage. This stage puts the z signal through the threshold detector. The threshold detector will include a threshold comparator to convert the z low pass filtered signal into a square pulse signal and a mapper. Symbol mapping helps convert a square pulse signal into a bit digital signal

Stage 3: From the 2 received I and Q bit sequences, we pass them through the mux multiplexer to restore the original signal.

III. Bit Error Rate.

To derive a minimum-error-probability receiver, rather than bit error, the criterion will be to find a receiver the minimizes the symbol error probability.
Expanding the received signal r(t) over the interval of T_s.

$r(t) = r_1\varphi_1(t) + r_2\varphi(t) + r_3\varphi_3(t) + \ldots$

Where $\varphi_1(t)$ và $\varphi_2(t)$ are determined as previous r_3 , r_4 … ~ $N(0, N_0/2)$
Based on the set of it is desired to make a decision as to the actual signal transmitted at the modulator output r_1, r_2, r_3.....Consider only the first projections, . The receiver is required to partition the m-dimensional space into four regions that achieves the minimum error probability.
We have:

Pr[error | s_1(t)] = Pr [error | s_2(t)] = Pr[error | s_3(t)] = Pr[error | s_4(t)]

So, the symbol error probality

$$P[\text{symbol error}] = 1 - [1 - Q(\sqrt{\frac{E_S}{N_0}})^2]$$

The number of trans symbols:	The symbol for decision:	The number of errored bits:
n_1 :00	n_{12} : 01	n_{12}
	n_{13} : 11	$2 \times n_{13}$
	n_{14} : 10	n_{14}
n_2 : 01	n_{21} : 00	n_{21}
	n_{23} : 11	n_{23}
	n_{24} : 10	$2 \times n_{24}$
n_3 : 11	n_{31} : 00	$2 \times n_{31}$
	n_{32} : 01	n_{32}
	n_{34} : 10	n_{34}
n_4 : 10	n_{41} : 00	n_{41}
	n_{42} : 01	$2 \times n_{42}$
	n_{43} : 11	n_{43}

$$P[\text{bit error}] = \frac{1}{T}\begin{pmatrix} n_{12} + 2n_{13} + n_{14} + n_{21} + n_{23} + 2n_{24} \\ +2n_{31} + n_{32} + n_{34} + n_{41} + 2n_{42} + n_{43} \end{pmatrix}$$

$$T = 2n_1 + 2n_2 + 2n_3 + 2n_4$$

The bit error probability of QPSK with a Gray mapping

$$P[\text{bit error}]_{QPSK} = Q\left(\sqrt{\frac{E_S}{N_0}}\right) = Q\left(\sqrt{\frac{2E_S}{N_0}}\right) = P[\text{bit error}]_{BPSK}$$

Here:

- **Eb** represents the energy per bit transmitted.
- **N0/2** denotes the one-sided power spectral density of the AWGN channel.

This equation tells us the probability of a bit error occurring during transmission.

IV. Conclusion.

- The bit error probability of QPSK is exactly the same as that of BPSK. However, the bandwidth of QPSK can be reduced by half compared to that of BPSK
- QPSK transmits two bits per symbol: Compared to BPSK (Binary Phase-Shift Keying) which transmits one bit per symbol, QPSK doubles the data rate for the same bandwidth.
- BER performance: For an Additive White Gaussian Noise (AWGN) channel, the theoretical BER of BPSK and QPSK are identical at a given Signal-to-Noise Ratio (SNR).
- Sensitivity to noise: While offering higher data rate, QPSK is slightly more susceptible to noise compared to BPSK. This is because noise can cause larger phase shifts, leading to a higher chance of mistaking one symbol for another.

V. Project 3

```
N0 = 10^-2;
EbN0_dB = 0:2:6; % EbNo simulink
EbN0 = 10.^(EbN0_dB /10);
Eb = EbN0 * N0; % The energy of one bit
Ntry = 5*10^3; % The number of transmitted bits
P_error_simul = zeros(1,length(EbN0_dB));
P_error_theo = zeros(1,length(EbN0_dB));
for j = 1:length(EbN0_dB)
%%
     ts = 1/1000;
     Tb = 1; % Time of 1 bit
     Ts = 2*Tb; % Time of 1 symbol
     Es = 2*Eb(j); %Energy of symbol
     V = sqrt(Es./Tb);
     Tc = Ts/10;
```

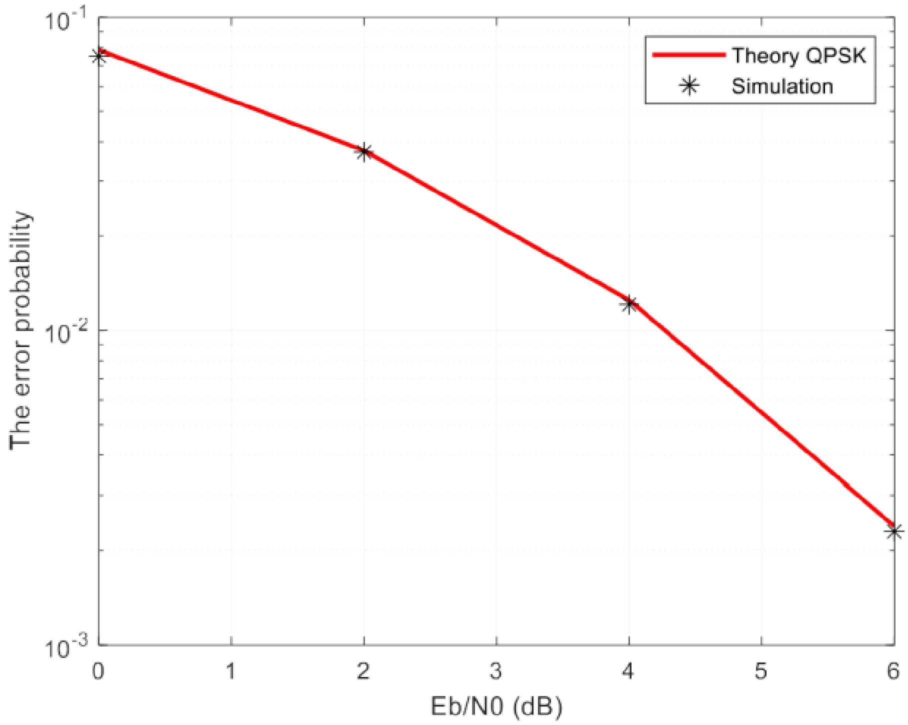
```
fc = 1/Tc;
t_1symbol = 0:ts:Ts-ts;
% Waveform
s1 = V*cos(2*pi*fc*t_1symbol); %00
s2 = V*sin(2*pi*fc*t_1symbol); %01
s3 = -V*cos(2*pi*fc*t_1symbol); %11
s4 = -V*sin(2*pi*fc*t_1symbol); %10

% ========
L = length(t_1symbol);
Bit = randsrc(1, Ntry, [0 1]); % Data
s = [];
t = [];
for i=1:2:Ntry
    if [Bit(i) Bit(i+1)] == [0 0]
       s = [s s1];
    elseif [Bit(i) Bit(i+1)] == [0 1]
       s = [s s2];
    elseif [Bit(i) Bit(i+1)] == [1 1]
       s = [s s3];
    elseif [Bit(i) Bit(i+1)] == [1 0]
       s = [s s4];
    end
t_isymbol = t_1symbol + i-1;
t = [t t_isymbol];
end

%% =============== AWGN channel
B = 1/ts;
N0_2 = N0./2;
Power_noise = N0_2*B;

w = sqrt(Power_noise)*randn(1,length(s));
% Received signal
r = s + w;
%% =============== Signal recovery
phi1 = s1./(sqrt(Es));
phi2 = s2./(sqrt(Es));
h1 = flip(phi1);
h2 = flip(phi2);
s11 = sqrt(Es); s12 = 0;
s21 = 0 ; s22 = sqrt(Es);
s31 = -sqrt(Es); s32 = 0;
s41 = 0; s42 = -sqrt(Es);
%% ============
Bit_rec = [];
for i = 1:Ntry/2
    Frame = r((i-1)*L + 1 : i*L); % Construct 1 Frame with L samples
of 1symbol
    y1 = conv(h1,Frame); % r(t) passes through the matched filter 1
    r1 = y1(L);
    y2 = conv(h2,Frame); % r(t) passes through the matched filter 2
    r2 = y2(L);
    d1 = (r1 - s11).^2 + (r2 - s12).^2; % The squared distance from r
to s1
```

```
        d2 = (r1 - s21).^2 + (r2 - s22).^2; % The squared distance from r
to s2
        d3 = (r1 - s31).^2 + (r2 - s32).^2; % The squared distance from r
to s3
        d4 = (r1 - s41).^2 + (r2 - s42).^2; % The squared distance from r
to s4
        % Comparator for decision
        if d1<d2 && d1<d3 && d1<d4
           Bit_rec = [Bit_rec 0 0];
        elseif d2<d1 && d2<d3 && d2<d4
           Bit_rec = [Bit_rec 0 1];
        elseif d3<d1 && d3<d2 && d3<d4
           Bit_rec = [Bit_rec 1 1];
        else
           Bit_rec = [Bit_rec 1 0];
        end
    end
    Bit_rec;
    % ================== The bit error probability
    % ------------- Simulation
    [Num, rate] = biterr(Bit, Bit_rec);
    P_error_simul(j) = rate;
    % ------------- Theory
    P_error_theo(j) = qfunc(sqrt(Es./N0));
end
P_error_simul;
P_error_theo;
figure(1)
semilogy(EbN0_dB, P_error_theo, 'r-', 'linewidth', 1.8); hold on;
semilogy(EbN0_dB, P_error_simul, 'k*', 'markersize',8);
xlabel('Eb/N0 (dB)'); ylabel('The error probability');
legend('Theory QPSK', 'Simulation')
grid on
disp('Done')
```

- Quadrature Phase Shift Keying (QPSK) communication system over an Additive White Gaussian Noise (AWGN) channel. Here's an explanation of each part of the code for your presentation:
- Initialization of Parameters:
- N0 = 10^-2;: This is the white noise power level.
- EbN0_dB = 0:2:6;: An array containing values of Eb/N0 in dB used for simulation.
- EbN0 = 10.^(EbN0_dB /10);: Conversion of Eb/N0 from dB to Eb/N0 ratio.
- Eb = EbN0 * N0;: Calculation of the energy of each bit.
- Setting Communication Parameters:

- Ntry = 5*10^3;: The number of bits transmitted.
- ts, Tb, Ts, Es, V, Tc, fc: Other parameters such as sampling time, bit and symbol duration, symbol energy, bit rate, carrier frequency, and symbol time.
- Variables s1, s2, s3, s4 represent waveforms corresponding to the QPSK symbols.
- Generating Random Data and Modulation:
- Bit = randsrc(1, Ntry, [0 1]);: Generating random data.
- Then, each pair of bits is modulated into one of the four QPSK symbols.
- Simulating the Transmission Channel:
- A Gaussian white noise channel is simulated by adding Gaussian noise to the transmitted signal.
- Recovering the Received Signal:
- Using matched filters to recover the modulated signal.
- Classification and Bit Error Calculation:
- Based on the distance between the received signal and the known QPSK symbols, deciding the transmitted bits.
- Comparing Simulation and Theoretical Results:
- Calculating the bit error rate and comparing it with theoretical results based on the Q-function of the normal distribution (qfunc).
- Plotting the Graph:
- Plotting the theoretical bit error rate (BER) curve against the actual BER curve to compare the performance of the system.

VI. Result



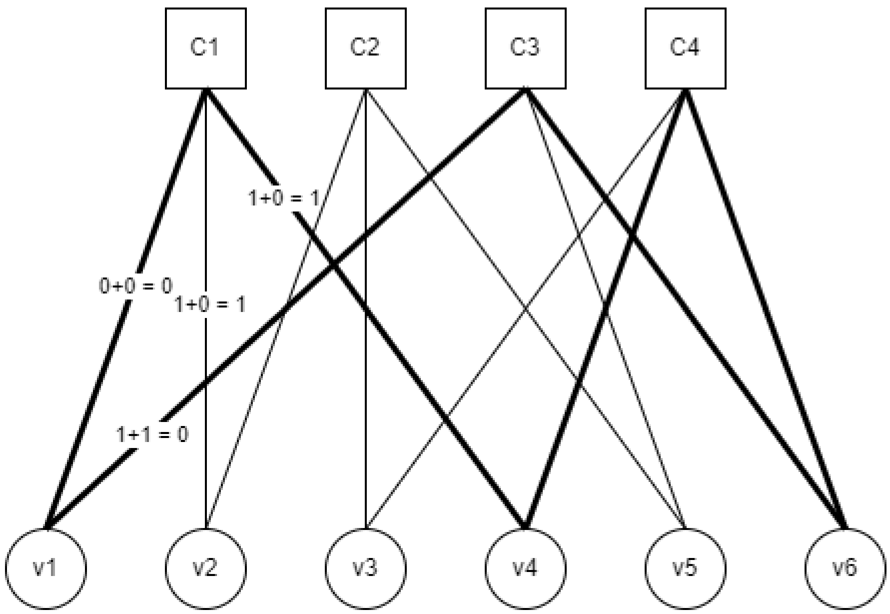
Theory QPSK
Simulation
The error probability
Eb/N0 (dB)
10^{-1}
10^{-2}
10^{-3}
0
1
2
3
4
5
6

Chapter 5: Error Correcting Codes

Parity Check Matrix H is given below

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Problem 1: Draw Tanner Graph for Parity Check Matrix?



Problem 2: Bit Flipping Algorithm

The transmitting bit string is c=[0 0 1 0 1 1], assum that the first bit is wrong. Use bit flipping algorithm to find $\hat{c}$.

So, bit string y = [1 0 1 0 1 1]

The first Check node send message to bit node v1, v2 and v4, so:

v1 = v2 + v4 = 0 + 0 = 0;

v2 = v1 + v4 = 1 + 0 = 1;

v4 = v1 + v2 = 1 + 0 = 1;

The second Check node send message to bit node v2, v3, v5:

v2 = v3 + v5 = 1 + 1 = 0;

v3 = v2 + v5 = 0 + 1 = 1;

v5 = v2 + v3 = 0 + 1 = 1;

Third Check node send message to bit node v1, v5 and v6:

v1 = v5 + v6 = 1 + 1 = 0;

v5 = v1 + v6 = 1 + 1 = 0;

v6 = v1 + v5 = 1 + 1 = 0;

The last Check node send message to bit node v3, v4 and v6:

v3 = v4 + v6 = 0 + 1 = 1;

v4 = v3 + v6 = 1 + 1 = 0;

v6 = v3 + v4 = 1 + 0 = 1;

A1[1; 3] = [0; 0] so the first bit node is flipped

A2[1; 2] = [1; 0] not fipped

A3[2; 4] = [1; 1] not fipped

A4[1; 4] = [1; 0] not fipped

A5[2; 3] = [1; 0] not fipped

A6[3; 4] = [0; 1] not fipped

The correct bit string: $\hat{c} = [0\ 0\ 1\ 0\ 1\ 1]$

Problem 3: Simulation Code

Parameters initialization

```
%Ma trận kiểm tra
H = [1 1 0 1 0 0
     0 1 1 0 1 0
     1 0 0 0 1 1
     0 0 1 1 0 1];

%Từ mã truyền
c = [0 0 1 0 1 1];

% Từ mã nhận
r = [1 0 1 0 1 1];

y = r;

%Số lần lặp
maxiter = 20;
iter = 0;

%Đánh dấu giải mã chưa thành công
success = 0;
```

Main loop with iteration and success flag conditions.

```
while iter < maxiter && ~success
    %Khởi tạo ma trận lỗi
    E = zeros(4, 6);
    for j = 1:4
        for i = 1:6
            if H(j, i) == 1
                %Xác định ma trận lỗi
                E(j, i) = mod(sum(y .* H(j, :)), 2);
            end
        end
    end
```

The Error Matrix when 2 for loop is done

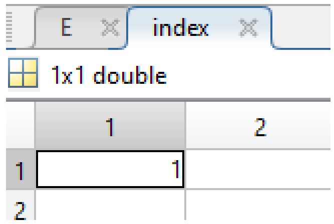
E

4x6 double

	1	2	3	4	5	6
1	1	1	0	1	0	0
2	0	0	0	0	0	0
3	1	0	0	0	1	1
4	0	0	0	0	0	0

```
    % Tìm vị trí có nhiều lỗi nhất
    for i = 1:6
        M(i) = sum(E(:, i));
```

```
    end
    [~, index] = max(M);
```
Colunm has the most error




```
    % Sửa lỗi
    if M(index) ~= 0
        y(index) = mod(y(index) + 1, 2);
    end

    % Kiểm tra sau khi đảo bit
    areErrorsPresent = check_errors(H, y);
    if areErrorsPresent == 0 % Không lỗi
        success = 1;
        disp("No error");
    else %Có lỗi
        disp("Still errors");
    end
    iter = iter + 1;
end
```
Algorithm of this code:

Main loop, break when reach to maxiter or success.

- Create Error Matrix E
- If H(j, i) = 1, calculate sum of product of y and row j of H
- E(j, i) equal to modulo – 2 of SOP
- Find the column has most error and save in variable “index”
- Use modulo – 2 of y(index) to flipped this [index] bit of y
- Use “check_errors” function to check error
- If no error then success = 1 and break this loop, else increase iter and back to the first step.

Appendix

Ordinal Number	Student Name	Student ID	Task	Complete	Point
01	Pham Hoai An (Team leader)	21207120	Coding, report, slide powerpoint project 4	100%	
02	Huynh Thi Ngoc Phuc	21207195	Coding, report, slide powerpoint project 3	100%	
03	Do Minh Chuong	21207126	Coding, report, slide powerpoint project 1	100%	
04	Tran Thien Phuc	21207077	Coding, report, slide powerpoint project 2	100%	