

Báo cáo: Xử lý tín hiệu số
NHÓM 14

Chủ đề: Thiết kế mô phỏng bộ điều chế và giải điều chế IQ với điều chế 16QPSK

Thành viên:

Nguyễn Minh Đức: B21DCCN249

Hoàng Xuân Đức: B21DCCN033

Lương Văn Luyện: B21DCCN504

Lê Đình Hưng: B21DCCN423

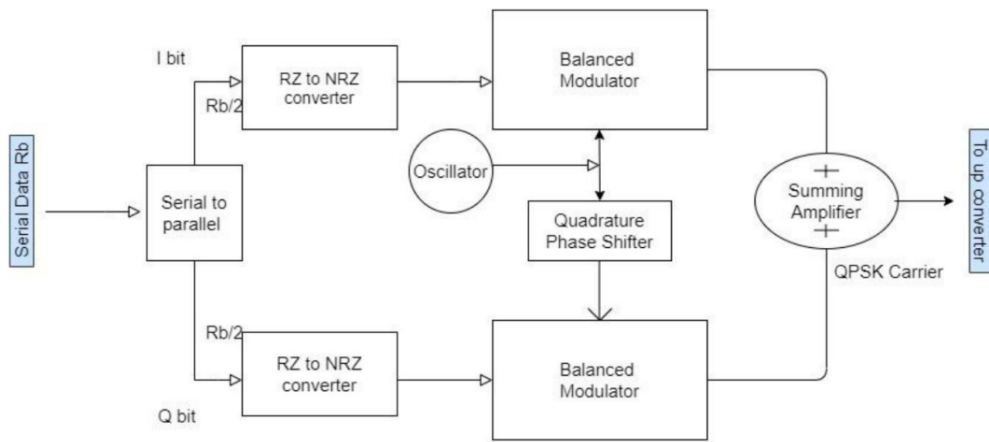
1. Sơ đồ Khối

Bộ điều chế QPSK:

Bộ điều chế QPSK bao gồm hai bộ điều chế khóa dịch pha nhị phân (BPSK), nối tiếp với thanh ghi dịch chuyển đổi song song, bộ tạo dao động và bộ dịch pha 90° . Chuỗi bit nối tiếp nhị phân với tốc độ bit R_b được áp dụng cho bộ điều chế được chuyển đổi thành chuỗi

song song hai bit I-bit và Q-bit, mỗi bit có tốc độ bit $R_b/2$. Các bit I và Q này được áp dụng cho các bộ điều chế BPSK, có tần số sóng mang trực giao với nhau. Tín hiệu trực giao này đạt được thông

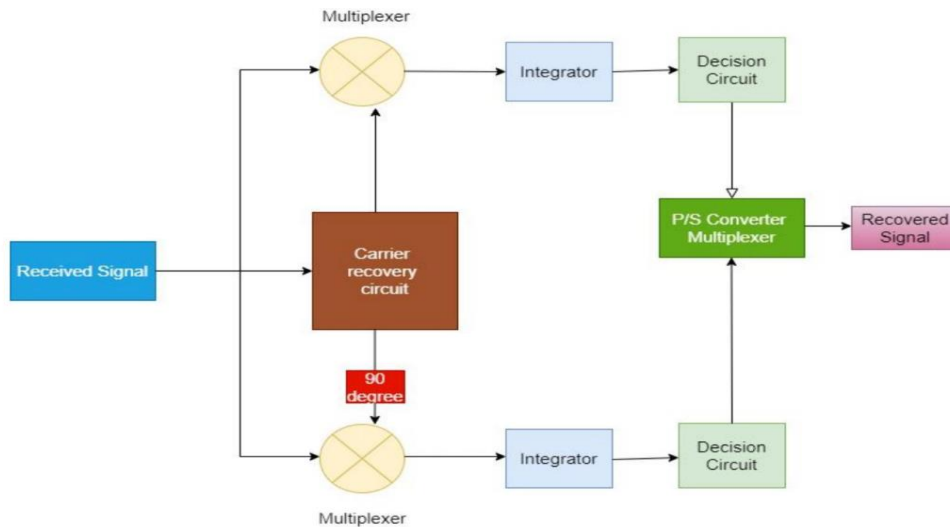
qua bộ dịch pha; làm thay đổi tín hiệu được áp dụng thành 90° . Đầu ra từ cả hai bộ điều chế BPSK được thêm vào bởi bộ khuếch đại tổng hợp; dẫn đến tín hiệu điều chế QPSK



Block diagram of QPSK Modulator

Bộ giải điều chế QPSK:

- Thiết kế mạch khôi phục sóng mang nhất quán cho giải điều chế các tín hiệu PSK sóng mang bị triệt tiêu là rất quan trọng và liên quan đến một số cân nhắc và đánh đổi hiệu suất. Tín hiệu điều chế số (QPSK) được đưa đến bộ giải điều chế. Và nó được áp dụng cho bộ trộn I, được truyền động với pha sóng mang 0° và cho bộ trộn Q với pha sóng mang 90° . Mạch khôi phục sóng mang tái tạo tham chiếu nhất quán cho giải điều chế và được định tuyến đến bộ điều biến cân bằng (bộ nhân). Bộ nhân trích xuất các luồng dữ liệu trong pha (I) và pha cầu phương (Q), được lọc thông thấp và cấp cho phần cảm tương ứng của bộ chuyển đổi NRZ của bộ đồng bộ hóa bit và bộ điều hòa tín hiệu



QPSK là phương thức truyền dữ liệu số qua tín hiệu tương tự analog. Để đạt được điều đó, các bit Dữ liệu được nhóm thành các cặp và mỗi cặp được đại diện bởi một dạng sóng cụ thể, được gọi là một biểu tượng, sẽ được gửi qua kênh sau điều chế sóng mang.

→ Các tín hiệu QPSK được định nghĩa toán học là

$S_i(t) = A \cos(2\pi f_c t + \theta_i)$	$0 \leq t \leq T$
--	-------------------

Trong đó: $\theta_i = (2i - 1)\pi/4$

Ta sẽ chia nó thành:

$$s_t(t) = A[(\cos \theta_i) * (\cos 2\pi f_c t)] - A[(\sin \theta_i) * (\sin 2\pi f_c t)]$$

$$s_t = s_{i1}\phi_1(t) + s_{i2}\phi_2(t)$$

Trong đó

$\phi_1 = \sqrt{2/T} \cos 2\pi f_c t,$ $\phi_2 = -\sqrt{2/T} \sin 2\pi f_c t,$ $s_{i1} = \sqrt{2} \cos \theta_i$ $s_{i2} = \sqrt{2} \sin \theta_i$	$0 \leq t \leq T$ $0 \leq t \leq T$
---	--

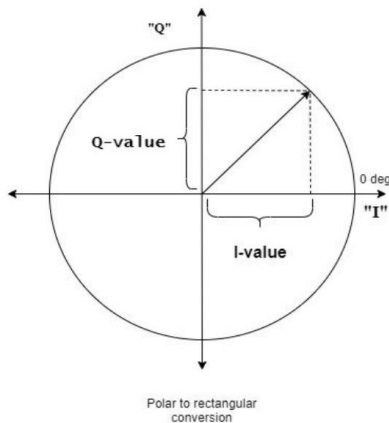
$s E \sin \theta$ $i_2 = \sqrt{i}$	
------------------------------------	--

$$\theta_i = \tan^{-1}(s_{i2}/s_{i1})$$

Ta thu được phương trình cuối cùng là:

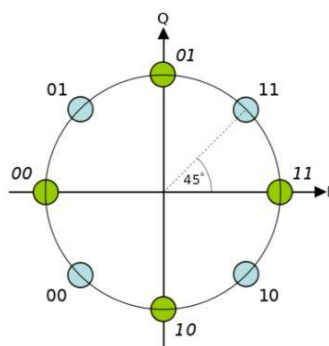
$$S(t) = A/\sqrt{2} [I(t)\cos(2fct) - Q(t) \sin(2fct)] \text{ trong đó } -\infty < t < \infty$$

Sơ đồ Constellation của điều chế QPSK trông như sau:



Ở đây, giá trị Q có nghĩa là thành phần cầu phương và giá trị I có nghĩa là cùng pha thành phần

Quadrature phase-shift keying (QPSK)



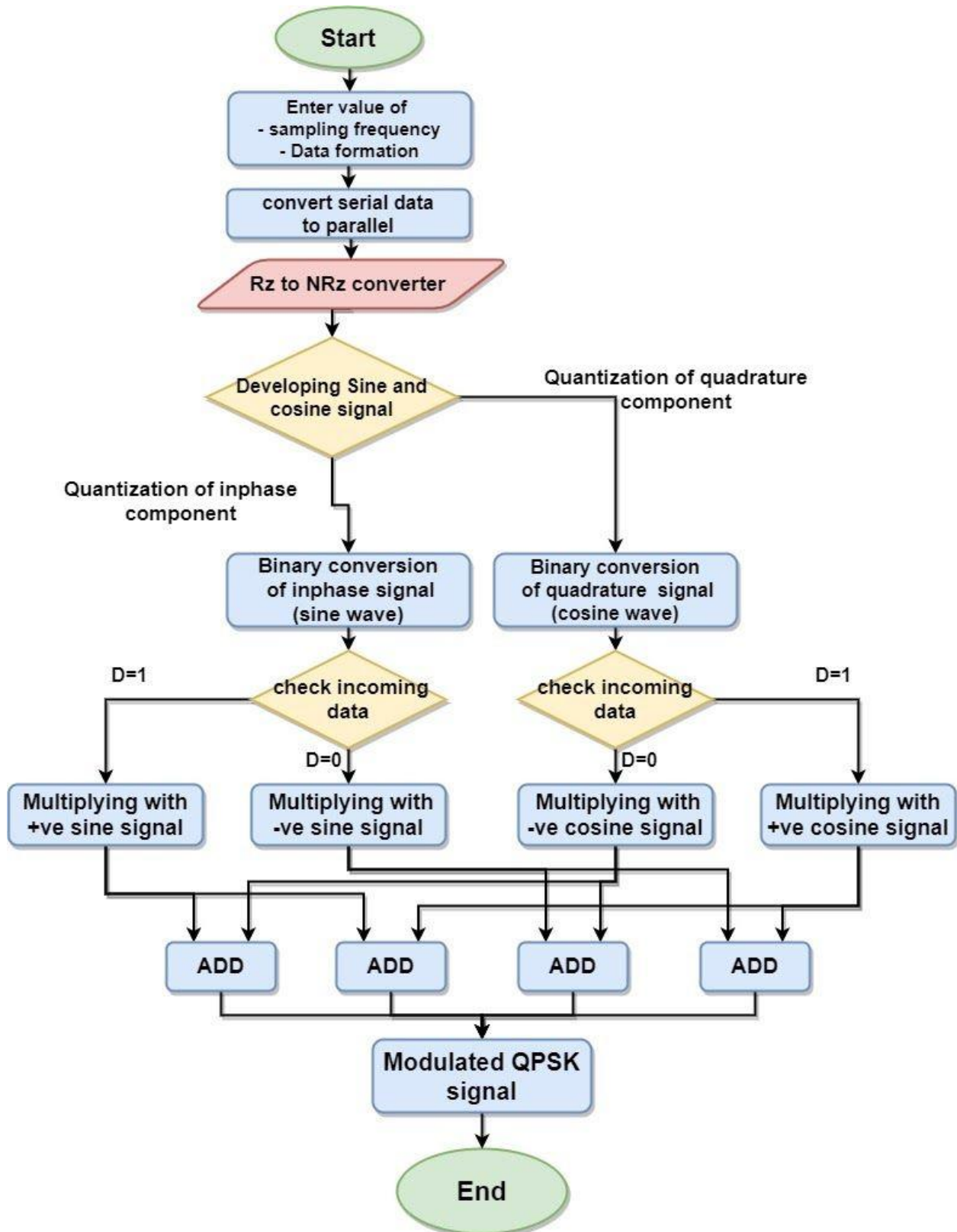
Phase of QPSK	2bits Inputs
$\pi/4$	10
$3\pi/4$	00
$5\pi/4$	01
$7\pi/4$	11

Figure : Constellation diagram for QPSK with Gray coding. Each adjacent symbol only differs by one bit.

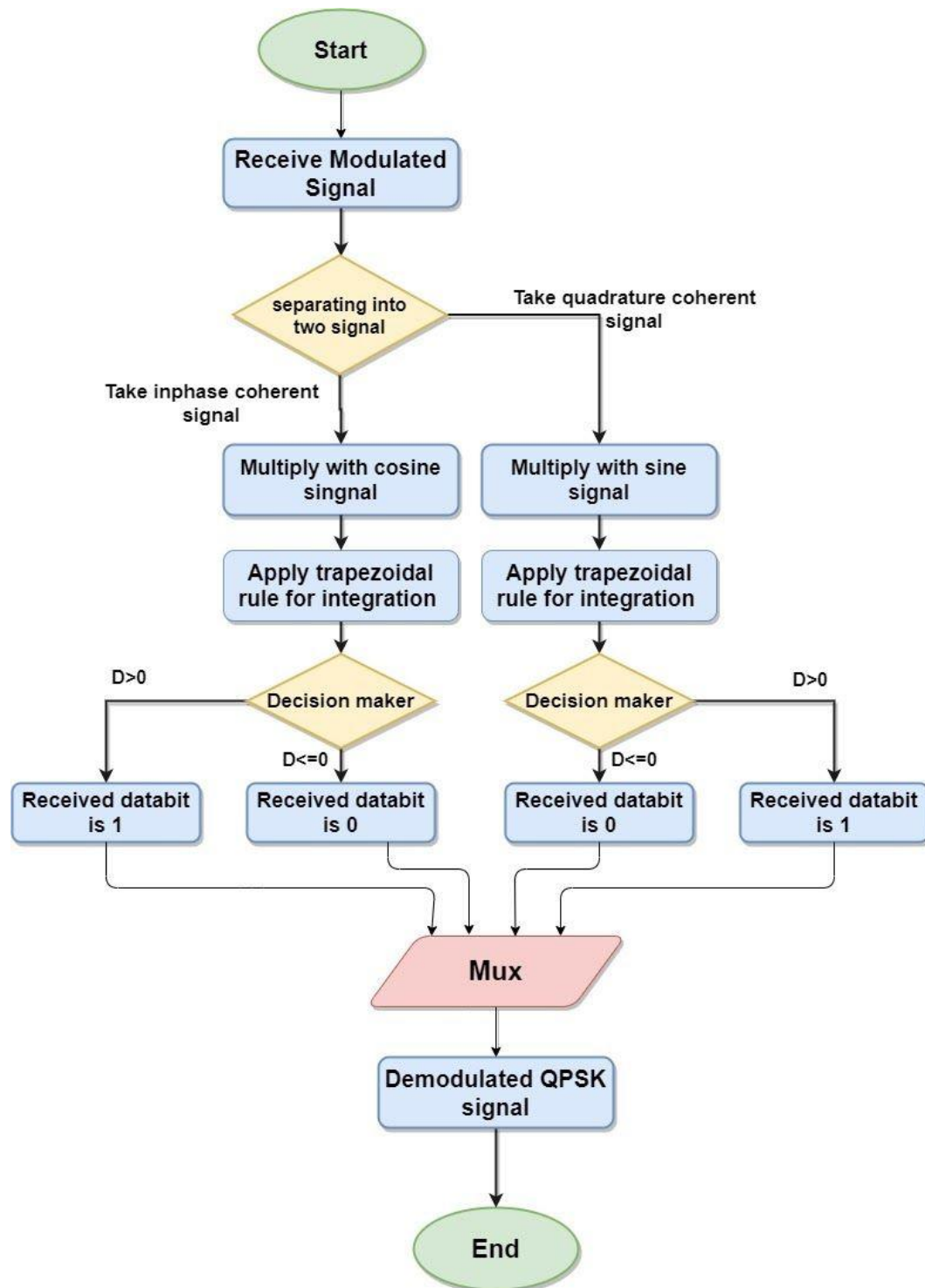
Thông số kĩ thuật của code Matlab

1. Điều chế QPSK
2. Tần số sóng mang: 10MHz
3. Tốc độ dữ liệu: 1024 bps
4. Loại nhiễu: AWGN (white Gaussian noise)
5. Tỷ lệ SNR (tỉ lệ tín hiệu/ tạp âm) : 10
6. Phương pháp phân tích: Trapezoidal rule

Sơ đồ



Bộ giải điều chế:



Code:

```
clc;
close all;
cnt=0;
data = randn(50,1);
for i=1:length(data)
    if data(i)<=0
        data(i)=0;
    elseif data(i)>0
        data(i)=1;
    end
end
figure(1)
stem(data, 'linewidth',3), grid on; %plotting the data which is gonna send.
title(' Original Data ', 'FontSize',15);
xlabel("data");
ylabel("amplitude");
axis([ 0 length(data) 0 1]);
data_NZR = 2*data-1; % Data Represented at NZR form for QPSK modulation
s_p_data = reshape(data_NZR,2,length(data)/2); % S/P conversion of data
br = 10.^6; %Let us transmission bit rate 1000000
f = br; % minimum carrier frequency
T = 1/br; % bit duration
t = T/99:T/99:T; % Time vector for one bit information
%%%% QPSK modulation %%%%
y=[];
y_in=[];
y_qd=[];
y_noise = [];
for i=1:length(data)/2
    y1=s_p_data(1,i)*cos(2*pi*f*t); % inphase component
    %disp("size of y1: "+size(t));
    Y2 = s_p_data(2,i)*sin(2*pi*f*t) ;% Quadrature component
    Y_
    in = [y_in y1]; % inphase signal vector
    Y_
    qd = [y_qd y2]; %quadrature signal vector
    noise = awgn(i,10,'measured'); %noise component
    9
    %noise= s_p_data(1,i)*(sin(2*pi*f*t)+randn(1,99));
    %noise = ((sqrt(0.1)*(randn(1,99)+randn(1,99)))); %noise component
    y_noise = [y_noise noise]; % noise vector
    y=[y y1+y2+noise]; % modulated signal vector
    %disp(y);
end
%disp("size of the noise"+size(y_noise));
Tx_sig=y; % transmitting signal after modulation
tt=T/99:T/99:(T*length(data))/2;
figure(2)
%disp("size of tt: "+size(tt));
subplot(4,1,1);
plot(tt,y_in,'linewidth',3), grid on;
title(' wave form for inphase component in QPSK modulation ', 'FontSize',12);
```



```

xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,2);
plot(tt,y_qd,'linewidth',3), grid on;
title(' wave form for Quadrature component in QPSK modulation ', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,3);
plot(y_noise,'linewidth',3), grid on;
title(' wave form for noise component in QPSK modulation ', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,4);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('QPSK modulated signal (sum of inphase and Quadrature phase signal and noise)', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
%%%% QPSK demodulation %%%%
Rx_data=[];
Rx_sig=Tx_sig; % Received signal
for(i=1:length(data)/2)
%% inphase coherent dictator %%
Z_in=Rx_sig((i-1)*length(t)+1:i*length(t)).*cos(2*pi*f*t);

```

Code:

```

clc;
close all;
cnt=0;
data = randn(50,1);
for i=1:length(data)
if data(i)<=0
data(i)=0;
elseif data(i)>0
data(i)=1;
end
end
figure(1)
stem(data, 'linewidth',3), grid on; %plotting the data which is gonna send.
title(' Original Data ', "FontSize",15);
xlabel("data");
ylabel("amplitude");
axis([ 0 length(data) 0 1]);
data_NZR = 2*data-1; % Data Represented at NZR form for QPSK modulation
s_p_data = reshape(data_NZR,2,length(data)/2); % S/P conversion of data
br = 10.^6; %Let us transmission bit rate 1000000
f = br; % minimum carrier frequency
T = 1/br; % bit duration
t = T/99:T/99:T; % Time vector for one bit information
%%%% QPSK modulation %%%%
y=[];
y_in=[];
y_qd=[];
y_noise = [];

```

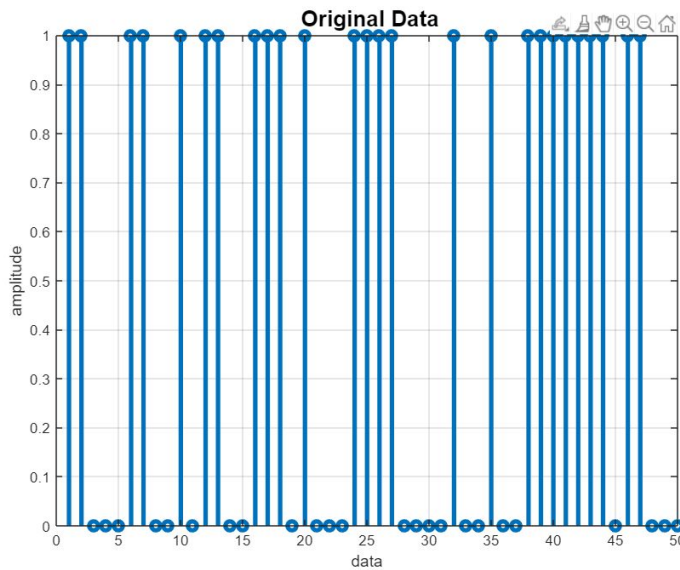
```

for i=1:length(data)/2
y1=s_p_data(1,i)*cos(2*pi*f*t); % inphase component
%disp("size of y1: "+size(t));
Y2 = s_p_data(2,i)*sin(2*pi*f*t) ;% Quadrature component
Y_
in = [y_in y1]; % inphase signal vector
Y_
qd = [y_qd y2]; %quadrature signal vector
noise = awgn(i,10,'measured'); %noise component
9
%noise= s_p_data(1,i)*(sin(2*pi*f*t)+randn(1,99));
%noise = ((sqrt(0.1)*(randn(1,99)+randn(1,99)))); %noise component
y_noise = [y_noise noise]; % noise vector
y=[y y1+y2+noise]; % modulated signal vector
%disp(y);
end
%disp("size of the noise"+size(y_noise));
Tx_sig=y; % transmitting signal after modulation
tt=T/99:T/99:(T*length(data))/2;
figure(2)
%disp("size of tt: "+size(tt));
subplot(4,1,1);
plot(tt,y_in,'linewidth',3), grid on;
title(' wave form for inphase component in QPSK modulation ', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,2);
plot(tt,y_qd,'linewidth',3), grid on;
title(' wave form for Quadrature component in QPSK modulation ', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,3);
plot(y_noise,'linewidth',3), grid on;
title(' wave form for noise component in QPSK modulation ', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
subplot(4,1,4);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('QPSK modulated signal (sum of inphase and Quadrature phase signal and noise)', "FontSize",12);
xlabel('time(sec)');
ylabel(' amplitude(volt0)');
%%%%% QPSK demodulation %%%%%
Rx_data=[];
Rx_sig=Tx_sig; % Received signal
for(i=1:1:length(data)/2)
%%% inphase coherent dictator %%%
Z_in=Rx_sig((i-1)*length(t)+1:i*length(t)).*cos(2*pi*f*t);

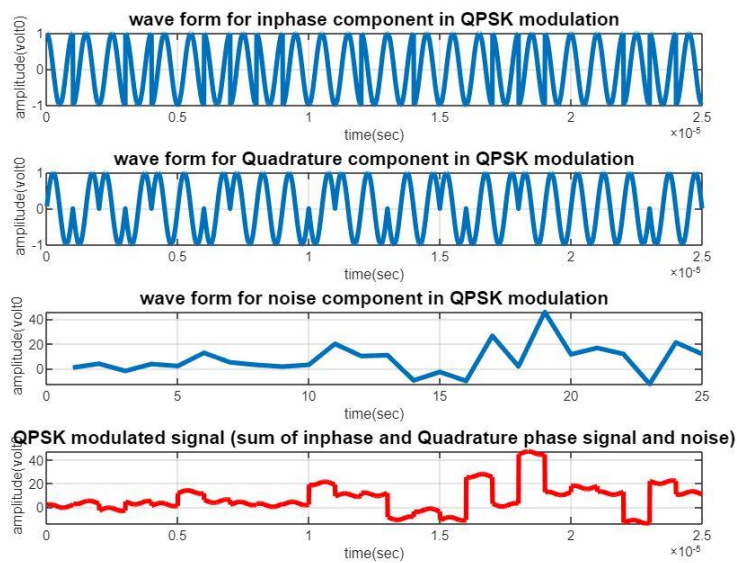
```

Kết quả:

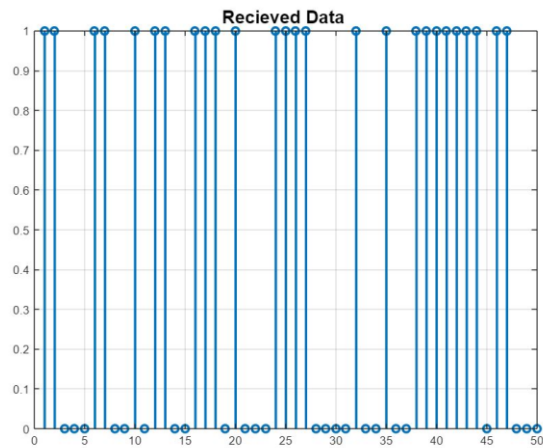
- Hiện thị dữ liệu đầu vào



- Hình thứ 2 cho thấy hai tín hiệu sóng mang, tín hiệu nhiễu và tín hiệu điều chế



- Hình thứ 3 cho thấy dữ liệu giải điều chế



- Giá trị của xác suất lỗi bit cho các phạm vi khác nhau (đối với AWGN):

0 -500	0 – 0.01
500 - 2000	0.01 – 0.2
2000 - 500000	0.2 – 0.3