**Communication Systems-2 (16EC62)**

**6th Semester**

**Assignment**

**2019-20**

**Title: OQPSK Modulation and Detection(MATLAB)**

|  |  |
| --- | --- |
| **Name** | **USN** |
| **CHANDANA** | **1RV17EC030** |
| **ANJANA MAHAVEER DAIGOND** | **1RV17EC015** |

**Submitted to**: Dr. Mahesh A, Associate Professor,

Electronics and Communication Engineering

**INTRODUCTION**

In a QPSK waveform,

* The carrier phase changes by ±180° whenever both the in-phase and quadrature components of the QPSK signal change sign.
* The carrier phase changes by ±90° whenever the in-phase or quadrature component changes sign.
* The carrier phase is unchanged when neither the in-phase component nor quadrature component changes sign.

Specifically, the 180° and 90° shifts in carrier phase can result in changes in the carrier amplitude (i.e., envelop of the QPSK signal) during the course of transmission over the channel, thereby causing additional symbol errors on detection at the receiver.

To mitigate this shortcoming of QPSK, we need to reduce the extent of its amplitude fluctuation. To this end we may use Offset QPSK. In this variant of QPSK, the bit stream responsible for generating the quadrature component is delayed (i.e., offset) by half a symbol duration interval with respect to in-phase component.

Accordingly, unlike QPSK, the phase transitions likely to occur in offset QPSK are confined to ±90°. However, ±90° phase transitions in offset QPSK occur twice as frequently but with half the intensity encountered in QPSK.

The amplitude fluctuations in offset QPSK due to filtering have smaller amplitude than in the case of QPSK.

**MATHEMATICAL EQUATIONS**

The two basis functions of offset QPSK are defined by,

The first basis function is exactly as that of for QPSK, but second basis function is different, i.e., it is delayed by half the symbol period.

**BLOCK DIAGRAM**

**MODULATION:**

a1(t)ttt9(99

Binary data sequence

Demultiplexer

Polar non-return to-zero level encoder

OQPSK signal

a2(t)

Delay

**DETECTION:**

Decision device (Threshold=0)

Integrator

Ф1(t)

Multiplexer

Estimate of

Transmitted

Sequence

Received signal

Decision device (Threshold=0)

I Integrator

Delay

ф2(t)

The distinguishing feature of the OQPSK transmitter is the block labelled demultiplexer. The function of the demultiplexer is to divide the binary wave produced by the polar NRZ-level encoder into two separate binary waves, one of which represents the odd-numbered dibits in the incoming binary sequence and other represents the even numbered dibits. The OQPSK transmitter may be viewed as two binary PSK generators with delay for quadrature components.

The functional composition of the OQPSK receiver is as follows:

1. Pair of Correlators, which have a common input. The two correlators are supplied with two locally generated orthonormal basis functions, which means the receiver is synchronized with the transmitter.
2. Pair of decision devices, which act on correlator outputs by comparing each one with a zero-threshold, here, it is assumed that the symbols 1 and 0 in the original binary stream at the transmitter input are equally likely, if x1>0, a decision is made in the favor of symbol one, if x1<0, a decision is made in the favor of symbol zero, similarly for quadrature channel.
3. Multiplexer, the function of which is to combine the two binary sequences produced by the pair of decision devices. The resulting binary sequence so produced is an estimate of the original binary stream.

**MATLAB CODE:**

clear all;

close all;

clc;

SNR = 0.09; %Signal to noise ratio

w=2\*pi\*5; %Frequency

t=0.01:0.01:0.2; %Time Vector

N=20; %Number of Bits

E\_s=1;

No = E\_s^2/SNR; %Noise Power

sig\_n = sqrt(No/2); %Noise standard deviation

data=floor(2\*rand(1,N));

data\_p=2\*data-1;

disp("Polar NRZ-level Encoder");

disp(data\_p)

data\_dem=reshape(data\_p,2,N/2);

disp("Odd and Even dibits")

disp(data\_dem)

y\_in=[];

y\_qd=[];

for i=1:N/2

y1=data\_dem(1,i)\*E\_s\*cos(w\*t); % inphase component

y2=data\_dem(2,i)\*E\_s\*sin(w\*t) ;% Quadrature component

y\_in=[y\_in y1]; % inphase signal vector

y\_qd=[y\_qd y2]; %quadrature signal vector

end

tt=0.01:0.01:(0.2\*N/2)+0.1;

y\_in1=[y\_in zeros(1,10)];

y\_qd1=[zeros(1,10) y\_qd];

y=[];

y=y\_in1+y\_qd1; % modulated signal vector

figure(1)

subplot(3,1,1);

plot(tt,y\_in1), grid on;

title(' waveform for inphase component in OQPSK modulation ');

xlabel('time(sec)');

ylabel(' amplitude(volt)');

subplot(3,1,2);

plot(tt,y\_qd1), grid on;

title(' waveform for Quadrature component in OQPSK modulation ');

xlabel('time(sec)');

ylabel(' amplitude(volt)');

subplot(3,1,3);

plot(tt,y), grid on;

title('OQPSK modulated signal (sum of inphase and Quadrature phase signal)');

xlabel('time(sec)');

ylabel(' amplitude(volt)');

n=sig\_n\*randn(1, length(y));

Rx\_sig=[];

Rx\_sig=y+n;

figure(2)

plot(tt, y);

hold on;

plot(tt, Rx\_sig);

grid on;

title('Transmitted and recieved signals');

legend('Transmitted waveform','Recieved waveform');

sigma1 = E\_s\*cos(w\*t); %Create basis functions

sigma2 = E\_s\*sin(w\*t);

index = 1:length(t);

for ii=0:N/2-1 %detect the signal

X(ii+1) = sum(sigma1.\*Rx\_sig(ii\*length(t) + index));

Y(ii+1) = sum(sigma2.\*Rx\_sig(ii\*length(t) + index+10));

end

currFig = figure; %double buffer so window

set(currFig,'DoubleBuffer','on'); %does not flash

title('Detections and decision regions for OQPSK') ; %Plot the detections

hold on;

axlim = 30;

axis([-axlim axlim -axlim axlim]);

plot([0 0],[-axlim axlim],[-axlim axlim],[0 0]);

rx=[];ry=[];

for ii=1:N/2 %Make decisions on received bits

if(X(ii) < 0&& Y(ii)<0)

rx(ii)=-1;ry(ii)=-1;

elseif (X(ii) < 0&& Y(ii)>0)

rx(ii)=-1;ry(ii)=1;

elseif (X(ii) > 0&& Y(ii)<0)

rx(ii)=1;ry(ii)=-1;

else

rx(ii)=1;ry(ii)=1;

end

figure(currFig);%Constellation diagram

if (and((rx(ii)-data\_dem(1,ii)),1)||and((ry(ii)-data\_dem(2,ii)),1))

plot(X(ii),Y(ii),'ro');

else

plot(X(ii),Y(ii),'bx');

end

hold on

pause(0.1);

end

xe=rx-data\_dem(1,:);

ye=ry-data\_dem(2,:);

q=0;

for i=1:N/2

a=and(xe(i),1);

b=and(ye(i),1);

if(a||b)

q=q+1;

end

end

disp('Number of Errors');

disp(q);

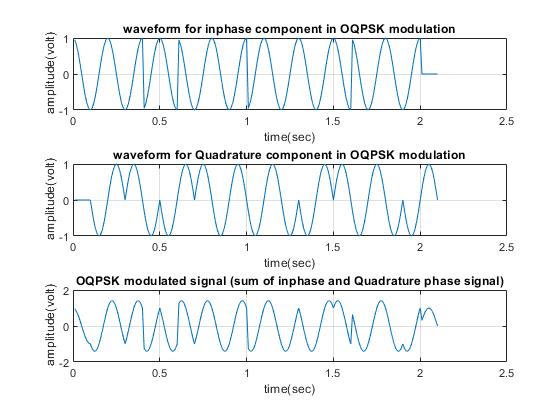
%Calculate the percentage error

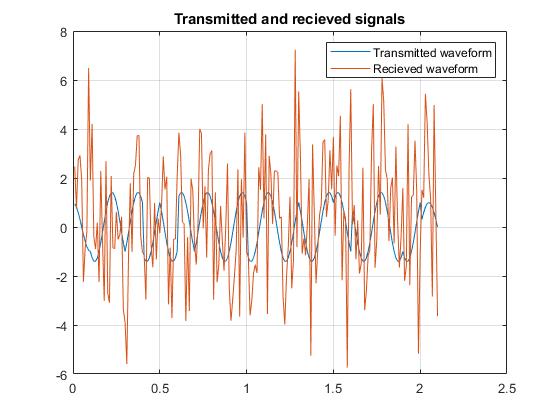
p\_error =( 2\*q / N )\*100;

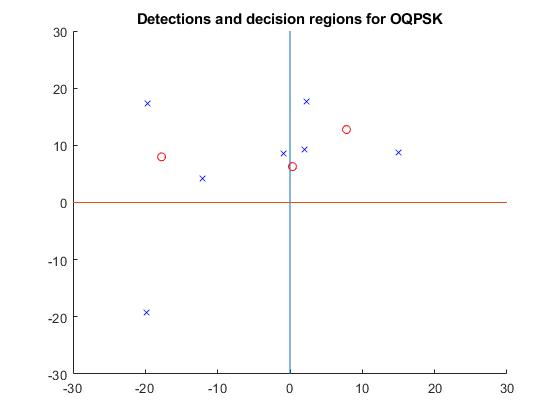
disp('Probability of error in %')

disp(p\_error);

**RESULTS:**

****





SNR=0.09

Number of Errors

3

Probability of error in %

30