QPSK Modulation & Demodulation
Project No. 3

Instructor: Dr. Mohammad Jubran

Name: Haitham Da'ana

ID: 1121331

Section: 10-11AM

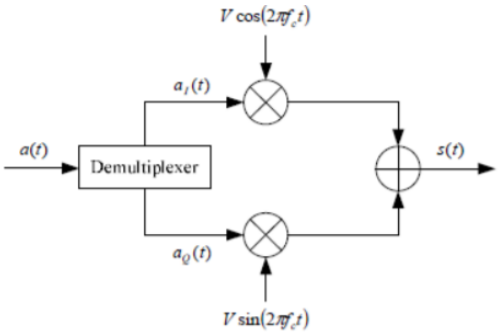
I. Introduction:

In order to send and transmit more data QPSK Modulation was introduced to help transmit more data over channel at lower bandwidth, the main idea about QPSK is to divide the transmitted bits into two segments, one is ***quadrature***, and the other is ***inphase***. The odd (odd bits are determined by the sequence and position of the oncoming input bits) bits by convention are separated and called ***quadrature*** bits, same goes with ***inphsae*** bits which are the even bits, these bits are equal in amplitude (same energy is used to transmit either one odd or even), but have different phase angles.

II. Theory:

1. Transmitter:

As mentioned before in the introduction, the bits are divided into two segments, one segment is called the inphase and the quadrature and then they are loaded onto the same carrier frequency but at different phase angles, see the figure below.

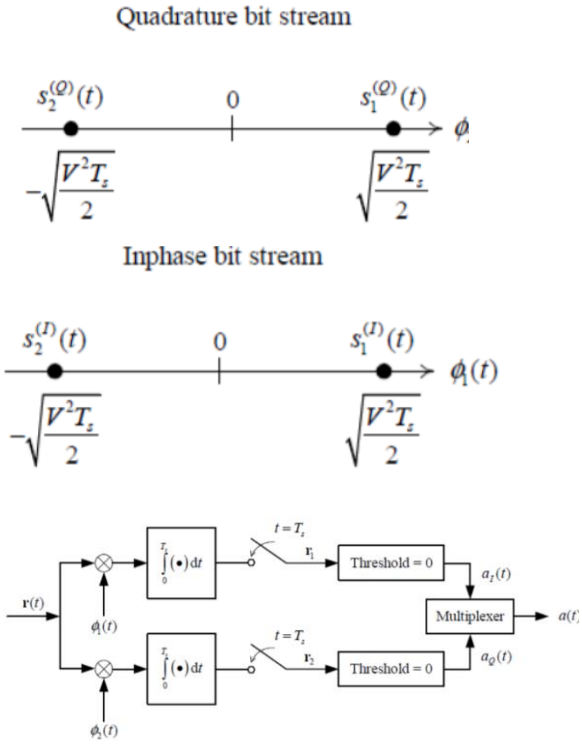


Figure(1): QPSK Modulator

They are partitioned by a multiplexer, then loaded onto a carrier signal then into a summer, and through an antenna or a wire or any transmission medium, they are distorted by additive white Gaussian noise which is added to the original signal.

2. Reciever:

At the receiver the signal is collected and then directed and multiplied by two carrier frequencies one with phase shift of 0 and one with 90 degrees phase shift, the output is then directed to an integrator with integration period of 0 to Ts, the result is r_1, and r_2, then they are directed to an amplitude shift keying detector which detects the amplitude of the signal and decides the received quadrature and inphase bit. See the figure below:



Figure(2): QPSK Demodulation

Bits then enter multiplexer for final sorting process.

III. Procedure of Matlab Simulation of QPSK:

1. Stage one: Bits Generation
   At first we generated about one million random bits, using randn, matrix was generated in an array of size (N/2,1), the quadrature bits are then multiplied by with j, and inphase are left to reperesent the real side part of the transmitted signal, the transmitted signal is sum of quadrature and inphase bits and the noise

2. Stage two: Reciever
   At the receiver side, a simple comparator was used to detect the bits of the signal, using simple two comparators, the first comparator compares the real part of the signal with threshold of 0, if it's bigger than 1 is decided, 0 it's below 0, same thing with the imaginary part. The signal is then represented by zero's and ones.

3. Stage Three: Error Calculations
   The randomly generated signal is then compared with the received signal using a simple for loop to check for each entry of the TxBits array and the RxBits array, if it detects difference in bits, it increments the error counter, which is initialized to be zero at first.

4. Stage Four: Results
   The matlab code was then made into a function of signal voltage V, it calculated the Symbol and Bit Error Rate as function of V, assuming that the Power in the noise is assumed to be 1.

   Voltage was calculated as following:

   (Signal-to-Noise Ratio)dB was made to be an array 0 to 10, Voltage was calculated as shown below from the value of SNRdb, Voltage was then obtained for each value of SNRdb, and voltage of signal was calculated and put in an array as following:

   $$SNRdb = 10log10(SNR)$$

   $$SNR = 10^{\frac{SNRdb}{10}}$$

   $$P_s = 10^{\frac{SNRdb}{10}}$$

   $$V = \sqrt{10^{\frac{SNRdb}{10}}}$$

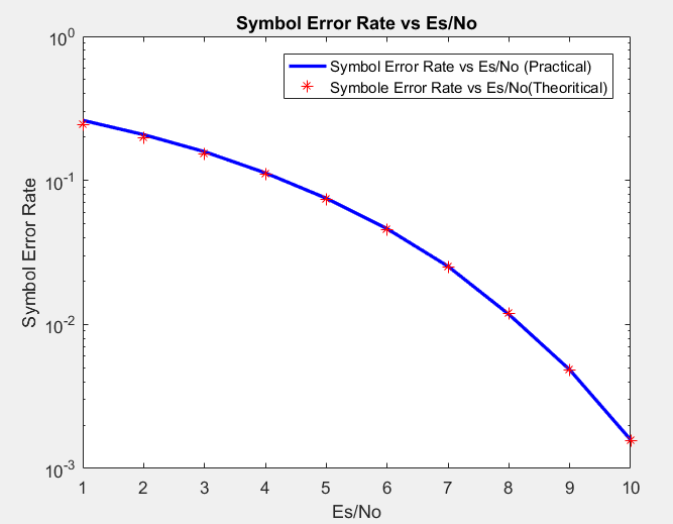
   The results were Symbol Error Rate & Bit Errror Rate drawn as a function of SNRdb.

   The theoretical Symbol Error Rate of QPSK, is given by this equation:

   $$P_{QPSK} = erfc\left(\sqrt{\frac{E_s}{2N_0}}\right) - \frac{1}{4}erfc^2\left(\sqrt{\frac{E_s}{2N_0}}\right)$$

IV. Results

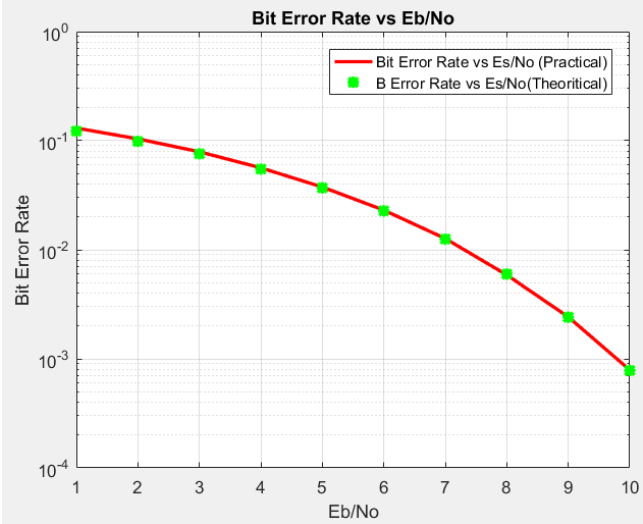
The probability of symbol error of QPSK that was obtained experimentally through the code and the actual theoretical one was drawn and plotted on top of it for accuracy comparison as shown in figure down below:



Figure(3): SER vs SNR(dB)

It can be noted that the results obtained are accurate and precise for higher SNR ratios, and that there was a deviation at Es/No=1 that is because at lower values of SNRdB the value of bit error is high.

The probability of bit error rate of QPSK was obtained plotted, and the theoretical one was also plotted on the same figure as can be shown below:



V. Conclusion

It can be noted that our results came accurate and agrees with the theoretical ones, and representing signals as imaginary and real helped us a lot in simulation process, without getting involved in time domain complications.

Appendix:

Matlab Code:

```
%%%% Transmitter %%%%
function BER = qpsk(v)
i=1;
N=2000000;
SNRdb=1:1:10;
Rb=1000;
Tb=(1/Rb);
Ts=Tb*2;
rN=rand(1,N)';
TxBits=round(rN)';
fc=Rb;
sq=[];
si=[];
TxBits_p=2*TxBits-1; %
for i=1:N
    if mod(i,2)==0
        e_bit=TxBits_p(1,i);
        si=[si e_bit];
    else
        o_bit=TxBits_p(1,i);
        sq=[sq o_bit];
    end
end
s=((si+j*sq)*1/sqrt(2))*v;
w=((1/sqrt(2))*randn(N/2,1))+j*((1/sqrt(2))*randn(N/2,1));
r=s+(w');
s_i=sign(real(r));
s_q=sign(imag(r));

RxSymbols=r;

Rx_i=[];
Rx_q=[];
RxBits=[];

for i=1:N/2
    x=r(1,i);
    y=real(x);
    z=imag(x);
    if (y)>0
        Rx_i=1;
    else
        Rx_i=0;
    end
    if (z)>0
        Rx_q=1;
    else
        Rx_q=0;
    end
    RxBits=[RxBits  Rx_q Rx_i];
end
tempmatrix=zeros(2,N);
for i=1:N
    tempmatrix(1,i)=tempmatrix(1,i)+RxBits(1,i);
    tempmatrix(2,i)=tempmatrix(2,i)+TxBits(1,i);
end
```

```
compare=[];
compare=tempmatrix;
m=0;
for i=1:N
    u=compare(1,i);
    o=compare(2,i);
    if (u~=o)
        m=m+1;
    end
end
BER=m/N
SER=2*BER
end
```

Comparison Code

```
SNRdb=1:1:10;
rv=sqrt(10.^(SNRdb/10))
BERvsSNR=[];
BitER=[];
for i=1:1:10;
    v=rv(1,i)
    BERvsSNR=qpsk(v);
    BitER=[BitER BERvsSNR];
end

theorySer_QPSK = erfc(sqrt(0.5*(10.^(SNRdb/10)))) -
(1/4)*(erfc(sqrt(0.5*(10.^(SNRdb/10))))).^2;

SymbolER=2*BitER;
figure
semilogy(SNRdb,BitER,'-r','linewidth',2)
xlabel('Eb/No');
ylabel('Bit Error Rate');
title('Bit Error Rate vs Eb/No');
hold on;
semilogy(SNRdb, (theorySer_QPSK/2),'g*','linewidth',2);

legend('Bit Error Rate vs Es/No (Practical)','B Error Rate vs
Es/No(Theoritical)');
grid on;
hold off;
figure
semilogy(SNRdb,SymbolER,'-b','linewidth',2)
xlabel('Es/No');
ylabel('Symbol Error Rate');
title('Symbol Error Rate vs Es/No');
hold on;
semilogy(SNRdb, theorySer_QPSK,'r*','linewidth',1);
legend('Symbol Error Rate vs Es/No (Practical)','Symbole Error Rate vs
Es/No(Theoritical)');
hold off;
```

End of Document