Contents


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

EE558 MATLAB Project Andrew Jefferson 825333113

Problem 1

```
xdB = 0:1:10;
x = 10.^(xdB/10);
x1 = sqrt(x);
x2 = sqrt(x*2);
```

Problem 2

```
L = 100000;     % Bit length
n = 10;         % upsample amount

Eb = 1;         % bit energy
No = (10.^(([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])/10)).^-1;

Fc = 1000;      % carrier frequency
Tc = 1/Fc;      % carrier period

Fs = 10000;     % sample rate
Ts = 1/Fs;      % sample period

t1 = Ts:Ts:Ts*n*L;  % time
t1 = t1.';


for w = 1:length(No)    % main FOR loop
```

```
% random message
BPSK = randi([0, 1], L, 1);
BASK = randi([0, 1], L, 1);

noise = randn([L*n 1]);

for i = 1:L
   switch(BPSK(i))
       case 0
            BPSK(i) = -1;
       case 1
    end
end
```

```
% for i = 1:L
%     if(BASK(i) == 1)
%         BASK(i) = 2^0.5;
%     end
% end

% up sampling for BPSK
BPSKuS = zeros([L*n, 1]);
for i = 1:L
    switch(BPSK(i))
       case -1
           for j = 1:n
            BPSKuS(j + (i-1)*n) = -1;
           end
       case 1
           for j = 1:n
            BPSKuS(j + (i-1)*n) = 1;
           end
    end
end

% up sampling for BASK
BASKuS = zeros([L*n, 1]);
for i = 1:L
    if(BASK(i) == 0)
           for j = 1:n
            BASKuS(j + (i-1)*n) = 0;
           end
    end
    if(BASK(i) > 0)
           for j = 1:n
            BASKuS(j + (i-1)*n) = sqrt(2);
           end
    end
end

% BPSK transmitter

sig1 = BPSKuS .* cos(2*pi*Fc*t1);       % modulation

tsig1 = sig1 + sqrt(No(w)/2) * noise;   % noise added

% BPSK Reciever


rsig1 = tsig1 .* cos(2*pi*Fc*t1);        % demodulation

rsig1 = lowpass(rsig1, (n/(Fs/2)));     % low pass filter

rsig1 = rsig1(n/2:n:L*n-n/2);           % Sampling signal at bitrate

% BPSK Desision Threshhold
for i = 1:L
    if(rsig1(i) <= 0)
        rsig1(i) = -1;
    end
    if(rsig1(i) > 0)
        rsig1(i)  = 1;
    end

end
```

```
errorCount1 = 0;
% BPSK error
for i = 1:L
    if(BPSK(i) ~= rsig1(i))
        errorCount1 = errorCount1 + 1;
    end
end
BPSKerror(w, 1) = errorCount1/L;

%------------------------------------------------------------------------

%BASK transmitter

sig2 = BASKuS .* cos(2*pi*Fc*t1);       % modulation

tsig2 = sig2 + sqrt(No(w)/2) * noise;   % noise added

% BASK Reciever


rsig2 = tsig2 .* cos(2*pi*Fc*t1);       % demodulation

rsig2 = lowpass(rsig2, (n/(Fs/2)));     % low pass filter

rsig2 = rsig2(n/2:n:L*n-n/2);           % Sampling signal at bitrate

% BASK Desision Threshhold
for i = 1:L
    if(rsig2(i) <= sqrt(2)/4)
        rsig2(i) = 0;
    end
    if(rsig2(i) > sqrt(2)/4)
        rsig2(i)  = 1;
    end

end

errorCount2 = 0;
% BASK error
for k = 1:L
    if(BASK(k) ~= rsig2(k))
        errorCount2 = errorCount2 + 1;
    end
end
BASKerror(w, 1) = errorCount2/L;
```

Gray QPSK

```
Es = sqrt(2)/4;
inphase = zeros([L/2 1]);
quatrature = zeros([L/2 1]);

grayQPSK = randi([0,1], L, 1);
noise = randn([L*n 1]);

t2 = Ts:Ts:Ts*0.5*L*n;          % time
t2 = t2.';
```

```
% Gray Encoding of signal
for i = 1:L/2

   switch (grayQPSK(2*i - 1))
       case 0

           switch(grayQPSK(2*i))
               case 0
                   inphase(i, 1) = Es;
                   quatrature(i, 1) = Es;
               case 1
                   inphase(i, 1) = -Es;
                   quatrature(i, 1) = Es;
           end
       case 1

           switch(grayQPSK(2*i))
               case 1
                   inphase(i, 1) = -Es;
                   quatrature(i, 1) = -Es;
               case 0
                   inphase(i, 1) = Es;
                   quatrature(i, 1) = -Es;
           end
   end
end
% % up sampling for BPSK
inphaseUS = repelem(inphase, n*2);
quatratureUS = repelem(quatrature, n*2);

% Gray QPSK transmitter

sigI = inphaseUS .* cos(2*pi*Fc*t1);       % Inphase modulation
sigQ = quatratureUS .* sin(2*pi*Fc*t1);    % Quatrature modulation

QPSKsig = sigI + sigQ;                      % sum signal for transmit
tQPSKsig = QPSKsig + sqrt(No(w)/2) * noise;       % noise added



% QPSK Reciever

% inPhase portion
rQPSKsigI = tQPSKsig .* cos(2*pi*Fc*t1);       % demodulation

rQPSKsigI = lowpass(rQPSKsigI, (n/(Fs/2)));     % low pass filter

rinphase = rQPSKsigI(n:2*n:L*n-n);           % Sampling signal at bitrate

% quatrature portion
rQPSKsigQ = tQPSKsig .* sin(2*pi*Fc*t1);       % demodulation

rQPSKsigQ = lowpass(rQPSKsigQ, (n/(Fs/2)));     % low pass filter

rquatrature = rQPSKsigQ(n:2*n:L*n-n);           % Sampling signal at bitrate


% decision Treshold
for i = 1:L/2
    if(rinphase(i) > 0)
```

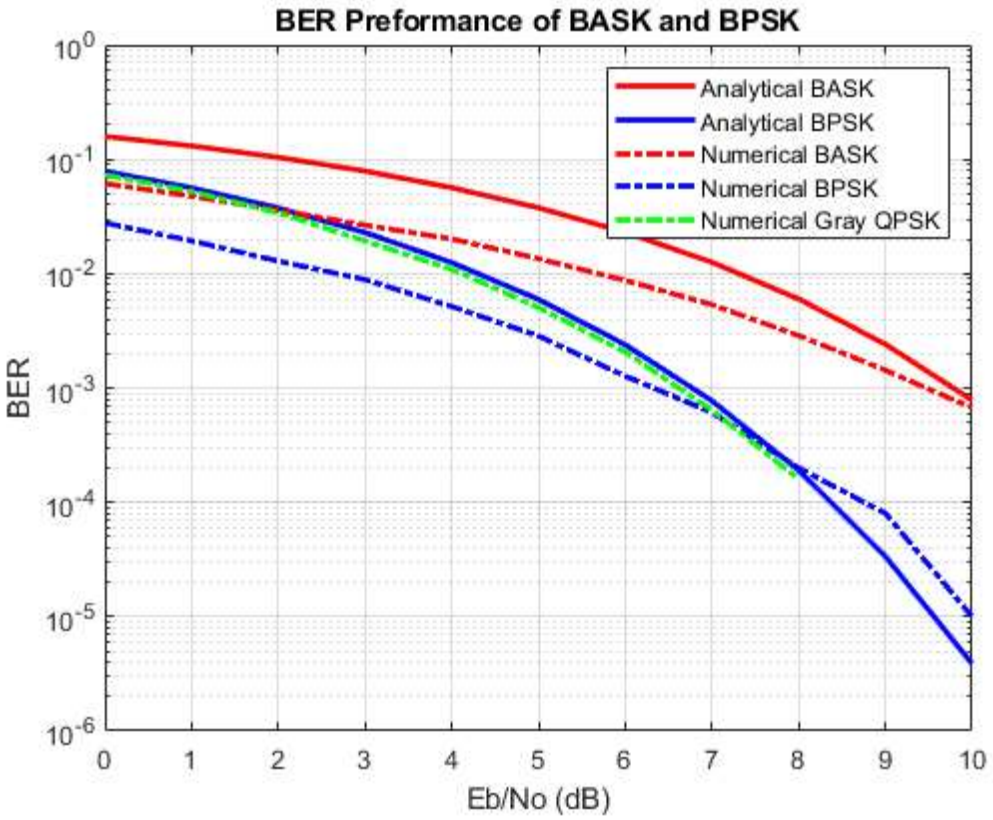
```
        if(rquatrature(i) > 0)
            rgrayQPSK(2*i-1) = 0;
            rgrayQPSK(2*i) = 0;

        else(rquatrature(i) <= 0);
            rgrayQPSK(2*i-1) = 1;
            rgrayQPSK(2*i) = 0;
        end

    else(rinphase(i) <= 0);
        if(rquatrature(i) > 0)
            rgrayQPSK(2*i-1) = 0;
            rgrayQPSK(2*i) = 1;

        else(rquatrature(i) <= 0);
            rgrayQPSK(2*i-1) = 1;
            rgrayQPSK(2*i) = 1;
        end
    end
end

errorCount3 = 0;
% QPSK error
for i = 1:L
    if(grayQPSK(i) ~= rgrayQPSK(i))
        errorCount3 = errorCount3 + 1;
    end
end
QPSKerror(w, 1) = errorCount3/L;
```

```
end % end of main FOR loop
```

BER Plot

```
BER = semilogy(xdB, qfunc(x1), xdB, qfunc(x2), ...
    xdB, BASKerror, xdB, BPSKerror, ...
     xdB, QPSKerror);

for i =1:5
    BER(i).LineWidth = 2;
end

BER(1).Color = 'red';
BER(2).Color = 'blue';
BER(3).Color = 'red';
BER(4).Color = 'blue';
BER(5).Color = 'green';
BER(3).LineStyle = '-.';
BER(4).LineStyle = '-.';
BER(5).LineStyle = '-.';

grid on;

legend('Analytical BASK','Analytical BPSK', ...
    'Numerical BASK', 'Numerical BPSK', ...
    'Numerical Gray QPSK');
title("BER Preformance of BASK and BPSK");
```

```
ylabel("BER");
xlabel("Eb/No (dB)");
```



Published with MATLAB® R2022a