Problem 2

- Main file used is Problem2Scriptv2G.
- All initial parameters are defined.
- In isiquant ,I tried to store all the convolution pulses generated from channel & rectangular pulses.
- Using formula Tsymb/Ts , I tried to determine the quantized value of hc. For the same, I have used the for loop which multiplies given hc with dirac delta function.

MATLAB CODE :

Main File :

```
%EE 5183: Foundations of Communications Exam Problem 2
clear all
clear classes;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SNR IN dB
%Students Can modify SNR Value and #BPSK Phasors generated
%Baseband Pulse
gsimcase=0;
stattype= 'statistical';
InfoNumStages=500;
Tsymb=1/128e3;
Ts=1/640e3;
p=Tsymb/Ts;
isiquant=[];
hq=[];
nr=1:1:5;
for n=1:1:p
    r=n*Tsymb;
    hq=[hq,r];
end
for n=1:1:p
l=conv(hq,rectangularPulse(n/Tsymb));
isiquant=[isiquant,l];
end
EbNodB=8;  % in dB
Es=1;
NumBitsPerSymbol=1;  %1-Tupple bits
SimCase = 'BPSK';
plot_baseband_figs= 'no'; %'yes', 'no' % affects class plots only
zfield=3;
h_unormal= [4.0825e-01, zeros(1,4), 8.1650e-01, zeros(1,4), 4.0825e-01];

%Baseband Equivalent Channel is typically complex
hc = h_unormal/norm(h_unormal);  %Normalized channel

NumStages=InfoNumStages+10;
binarysequence= [zeros(1,5), randi([0,1],1,InfoNumStages),zeros(1,5)];

NumSymbs=length(binarysequence)/NumBitsPerSymbol;
```

```
b=zeros(1, NumSymbs);
for k=1:NumSymbs
    % 1-tupple: b0 --> b0 mapping to complex coefficient
    switch binarysequence((k-
1)*NumBitsPerSymbol+1:k*NumBitsPerSymbol)*[1]';
    % BPSK Mapping Defined
        case 0
            b(k)=-1;
        case 1
            b(k)=1;
    end
end

axis([0,40,-1,1]);
%instantiate the object a:  "a" is of class BasbandeGenNew
%you must create a folder @BasebandGenNew and place class constructor and
p.rcrolloff=1.0;  %structure p
p.symbolrate=128e3;
p.NumSamplesTxPulse=21;
p.SampleLaunchPeriod=5;
p.Plotfig='no';
%p.hc=hc;                     % modification
a=BasebandGenNew(p, Es,b);  %constructor
%pulse has even symmetry;
matchedfilter=a.txpulse;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%DO NOT MODIFY HERE%%%%%%%%%%%%%%%%%%%%
%
basebandsig=a.basebandsig;
bkphasors=a.bkphasors;
Ts=a.Ts;
EbNoLin=10^(EbNodB/10);
nvar=(1/NumBitsPerSymbol)/EbNoLin;
isisig=conv(basebandsig,hc);
y=isisig+sqrt(nvar)*randn(size(isisig)); %ISI observation in AWGN is vector
y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ISIlength=conv(hc,a.txpulse);
L=length(ISIlength);
N=L*(Ts/Tsymb);
Nfinal=round(N);
hqn=[];
i=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
switch gsimcase   %conv(g(n), delta(n-d))
    case 0
       conv_g_delta=[zeros(1,20),1,zeros(1,20)].';  %size <=wlen+hlin-1
       wlen=31;
    case 1
       conv_g_delta=[0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,1,1,1,1,1,0,0,0,0,0].';
%size <=wlen+h-1
    case 2
       conv_g_delta=[0,0,0,0,0,1,0,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,0].';
%size <=wlen+h-1
       wlen=15;
    case 3
       conv_g_delta=(conv([0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
matchedfilter)).';
       wlen=25;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%h=quantize(ISIlength);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Equalizer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P=256;
dptr=length(hc)+wlen;
br=basebandsig.';

[eqobj, rhsvec]=AdvEqual(hc,br,conv_g_delta,wlen,P,dptr, nvar,stattype);
for k=1:1:length(hc)
  for t=1:1:N-1
     % hqn(t)=hc(t).*dirac(0-k*Tsymb);
  %    hqn(t)=
  end
end
figure(1)
subplot(2,1,1), plot(real(hc))
xlabel('sample index');
ylabel('Real');
title('Real Part of hc[n]');
subplot(2,1,2), plot(imag(hc))
xlabel('sample index');
ylabel('Imag');
title('Imag Part of hc[n]');

figure(2)
plot(a.txpulse)
xlabel('sample index');
ylabel('Amplitude');
title('Prototype Pulse');

figure(3)
subplot(2,1,1), plot(real(conv(hc,a.txpulse)))
xlabel('sample index');
ylabel('Real');
title('Real Part of ISI Pulse');
subplot(2,1,2), plot(imag(conv(hc,a.txpulse)))
xlabel('sample index');
ylabel('Imag');
title('Imag Part of ISI Pulse');

figure(4)
subplot(2,1,1), plot(real(basebandsig))
xlabel('sample time index');
ylabel('Real');
title('Real Part of Tx Complex Baseband Signal');
subplot(2,1,2), plot(imag(basebandsig))
xlabel('sample time index');
ylabel('Imag');
title('Imag Part of Tx Complex Baseband Signal');

figure(5)
subplot(2,1,1), plot(real(y))
xlabel('sample time index');
ylabel('Real');
title('Real Part of Received  Complex Baseband Signal: After Channel');
subplot(2,1,2), plot(imag(y))
xlabel('sample time index');
```

```
ylabel('Imag');
title('Imag Part of Received  Complex Baseband Signal: After Channel');

figure(8)
subplot(2,1,1), plot(real(isiquant));
subplot(2,1,2), plot(imag(isiquant));
%%%%%%%%%%%%%%%%%%%%%%%STUDENT CODE BELOW%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%Add a Method ViterbiEqualDetect to Class
BasebandGenNew%%%%%%%%%%
%We can alternatively add a function to the Basebandmodel if desired

% %ccc=ViterbiEqualDetect(arg1, arg2, ...argN);
% %numerrors=sum(xor(ccc.brecov, binary_sequence))
```

BasebandGen :

```
%EE5183: You do not neet to modify unless you wish to experiment
function[txpulseEs, basebandsig, bkphasors, Ts] =...
    BasebandGen(b, rcrolloff, symbolrate, NumSamplesTxPulse,
SampleLaunchPeriod,Es, plotfig)
% clear all
R=rcrolloff;  %rollof off of raised cosine pulse
Rsymb0=symbolrate; %Symbol Rate in kysymbols/second
Tsymb=1/Rsymb0;  %time duration in seconds between symbol launches
RATE=SampleLaunchPeriod; %Number of discrete samples between successive
symbol launches
OLEN=NumSamplesTxPulse;  %approximate value of the desired filter length
OLENP=RATE*( 2*ceil( ceil((OLEN-1)/RATE)/2))+1;  %minimum constraint filter
greater than OLEN
FLEN=(OLENP-1)/RATE +1;
N_T=(FLEN-1)/2;

Ts=(1/Rsymb0)/RATE;  %sampling resolution of pulse filter

FilterType='normal'; % raised cosine
Bp = rcosfir(R, N_T, RATE, Ts, FilterType);  %Matlab raise cosine
Bp = Bp/sum(Bp.^2); %normalize pulse energy to 1
txpulse=Bp;
%%%%%%%%%%%%%%%%%%%% Modifications done by me in this file
figure(7)
stem(txpulse);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=2^10; % FFT Size for analysis of puse frequency response
Fb=(1/Ts);


xaxf=((0:N-1)/N)*(Fb);
yaxfp=20*log10(abs(fft(Bp,N)));


Borig=Bp/10^(abs(yaxfp(1))/20);
%%%%%%%%%%%%%%%
%Use normalized pulse Energy
%
B=sqrt(Es)*Borig/sqrt(sum(Borig.*conj(Borig)));
txpulseEs=B;
```

```
%%%%%%%%%%%%%%%
if plotfig==1;
figure(1)
plot(1:2*N_T*RATE+1,B,'.');
xlabel('sample index')
ylabel('Raised Cosine Pulse Amplitude');
title('Discrete Time Domain Plot of Raised Cosine Pulse');


figure(2)
yaxf=yaxfp-yaxfp(1);
plot([xaxf-xaxf(end)/2], [yaxf(N/2+1:N),yaxf(1:N/2)], 'b');
xlabel('frequency in Hz');
ylabel('Frequency Spectrum Amplidtude in dB');
title('Frequency Plot of Raised Cosine Pulse');
end

b_dirac=upsample(b,RATE); %create bk*dirac function (see plot)
bkphasors=b_dirac;
svec=b(1)*B;  %Very 1st baseband pulse
yvec=conv(b_dirac, B); %sequence of baseband pulses
basebandsig=yvec;
xax0=[0:length(b_dirac)-1]*(1/Fb);
xax1=[0:length(svec)-1]*(1/Fb);
xax2=[0:length(yvec)-1]*(1/Fb);

if plotfig==1;
figure(3)
subplot(2,1,1), plot(xax1, real(svec),'.');
xlabel('sample time (sec)');
ylabel('Real');
title('1st Complex Baseband');
subplot(2,1,2), plot(xax1, imag(svec),'.');
xlabel('sample time (sec)');
ylabel('Imag');

figure(4)
subplot(2,1,1), plot(xax0, real(b_dirac),...
'--bs','LineWidth',0.1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','r',...
                'MarkerSize',3);
xlabel('sample time (sec)');
ylabel('Real');
title('Sequence of Complex Dirac Impulses');
subplot(2,1,2), plot(xax0, imag(b_dirac), ...
    '--bs','LineWidth',0.1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','r',...
                'MarkerSize',3);
xlabel('sample time (sec)');
ylabel('Imag');

figure(5)
subplot(2,1,1), plot(xax2, real(yvec),'.');
xlabel('sample time (sec)');
ylabel('Real');
title('Sequence of Complex Baseband Pulses');
subplot(2,1,2), plot(xax2, imag(yvec),'.');
xlabel('sample time (sec)');
```

```
ylabel('Imag');
end
```

BasebandGenNew :

```
classdef BasebandGenNew < handle
    %UNTITLED Summary of this class goes here
    %   Detailed explanation goes here
    %hinit, b, EbNodB, Es,
rcrolloff,symbolrate,NumSamplesTxPulse,SampleLaunchPeriod
    properties
        txpulse=[1, 2, 1]; %baseband pulse
        v=0.1;  % noise
        y=[];
        MappingType = 'QPSK';
        NFREQ=256;
        LupSample = 7;
        rcrolloff=[];
        symbolrate=[];
        NumSamplesTxPulse=[];
        SampleLaunchPeriod=[];
        Es=[];
        basebandsig=[];
        bkphasors=[];
        Ts=[];
        b=[];
        Plotfig='no';
     %   hc=[];           %modification
    end


    methods
        function obj=BasebandGenNew(varargin)

            for k=1:nargin
                switch k
                    case 1
                    obj.rcrolloff=varargin{1}.rcrolloff;
                    obj.symbolrate=varargin{1}.symbolrate;
                    obj.NumSamplesTxPulse=varargin{1}.NumSamplesTxPulse;
                    obj.SampleLaunchPeriod=varargin{1}.SampleLaunchPeriod;
                    obj.Plotfig=varargin{1}.Plotfig;
               %     obj.hc=varargin{1}.hc;
%modification
                    case 2
                        obj.Es=varargin{2};
                    case 3
                        obj.b=varargin{3};
                end
            end


[obj.txpulse, obj.basebandsig, obj.bkphasors, obj.Ts] =...
    BasebandGen(obj.b, obj.rcrolloff, obj.symbolrate,
obj.NumSamplesTxPulse, ...
    obj.SampleLaunchPeriod, obj.Es, obj.Plotfig);
```

```
        end
    end

end
```

AdvEqual Function :

```
function [ adveq, rhsvec ] = AdvEqual( varargin )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
for kindex=1:nargin
    switch kindex
        case 1
            adveq.himp = varargin{1};
            adveq.Nlength_h= length(adveq.himp);
            adveq.state_h    = zeros(1,adveq.Nlength_h-1);
        case 2
            adveq.b = varargin{2};
            adveq.Nlength_b= length(adveq.b);
        case 3
            adveq.gimp = varargin{3};
            adveq.Nlength_g= length(adveq.gimp);
            adveq.state_g    = zeros(1,adveq.Nlength_g-1);
        case 4
            adveq.weq     = 1;
            adveq.Nlength_w =varargin{4};
            adveq.state_w = zeros(1, adveq.Nlength_w-1);
        case 5
            adveq.P    = varargin{5};
        case 6
            adveq.np   = varargin{6};
        case 7
            adveq.nvar = varargin{7};
        case 8
            adveq.stattype = varargin{8};
        otherwise
            error('AdvEqual.m: Too many input arguments');
    end
end

adveq.hmat=zeros(adveq.Nlength_w,adveq.Nlength_w+ adveq.Nlength_h-1);
for nh=1:adveq.Nlength_w
    adveq.hmat(nh,nh:nh+adveq.Nlength_h-1)=adveq.himp.';
end

adveq.bmat=zeros(adveq.P,adveq.Nlength_h+adveq.Nlength_w-1);
for ng=0:adveq.Nlength_h+adveq.Nlength_w-2
    adveq.bmat(:,ng+1)=indorg(adveq.b, adveq.np-ng, adveq.np-ng+adveq.P-1);
end
```

```
adveq.Rvv=max(adveq.Nlength_w)*adveq.nvar*eye(max(adveq.Nlength_w));

zhead=zeros(ceil(0.5*(adveq.Nlength_w+adveq.Nlength_h-1-
adveq.Nlength_g)),1);
ztail=zeros(floor(0.5*(adveq.Nlength_w+adveq.Nlength_h-1-
adveq.Nlength_g)),1);
switch adveq.stattype
    case 'emperical'
        F1=(adveq.Rvv+
conj(adveq.hmat)*adveq.bmat'*adveq.bmat*adveq.hmat.');
        F2= conj(adveq.hmat)*adveq.bmat'*adveq.bmat*[zhead; adveq.gimp;
ztail];
        rhsvec=[zhead; adveq.gimp; ztail];
    case 'statistical'
        Rbbstat=adveq.P*var(adveq.b,1)*eye(adveq.Nlength_h+adveq.Nlength_w-
1);
        F1=(adveq.Rvv+ conj(adveq.hmat)*Rbbstat*adveq.hmat.');
        F2= conj(adveq.hmat)*Rbbstat*[zhead; adveq.gimp; ztail];
        rhsvec=[zhead; adveq.gimp; ztail];
    otherwise
        error('invalid statistical model')
end
tempval=inv(F1)*F2;
adveq.weq = tempval(1:adveq.Nlength_w);

adveq = class(adveq, 'AdvEqual');


end
```