```
%QPSK demodulator
%Group 11
%final version

clear all
close all
clc

showplots=0;                                    %enables/disables plots

fs=54000;                                       %Sampling frequency
fc=2100;                                        %Carrier freqeuncy
fbp1 =  2500 / (fs/2);
fbp2 =  1500 / (fs/2);
fbp=[fbp1,fbp2];
[pbp1,zbp1] = butter(8, fbp1);
[pbp2,zbp2] = butter(8, fbp2,'high');
% tic


%%input detector
powerin=0;
while powerin==0                                %a sample of sound is recorded
 sample=wavrecord(100,fs,1);                    %time is important, as it increased
 sample1 = filtfilt(pbp1, zbp1, sample);        %it is easier to detect the sound but
 sample2 = filtfilt(pbp2, zbp2, sample1);       %pilot might be missed
 powersample=sum(sample2.*sample2);             %threshold is 0.25 for comp10
 if powersample>=0.05                          %if the recorde signal is above threshold
     powerin=1;                                 %loop is finished and input detected! is dislpayed
     display('input detected!')                 %indicating that sound is detected
 end                                            %a high pass and band pass filter is applied
end                                             %with passband around 2100Hz
% toc                                             %standby time

% test = wavread('Team11_QPSK_sound.wav');      %for offline testing

test1=wavrecord(30*fs,fs,1);                    %record the detected input

tic

test2 = filtfilt(pbp1, zbp1, test1);            %a high pass and band pass filter is applied
test = filtfilt(pbp2, zbp2, test2);             %with passband around 2100Hz

t=[0:1:length(test)-1];                         %time vector is created



% figure
% pwelch(test);                                 %input in frequency domain



% Input Chopper
% Divides the input into two parts
% For the having different matrices for the two passwords
p=1;
p2=0;
p3=0;
p4=0;
ti=[0:1/fs:(1/fc)];                             %time vector for one carrier signal period


while p2==0                                     %looks for the silent part in between
if sum(test(p:p+99).*test(p:p+99))>0.003        %threshold is 0.04 for computer 10
        p=p+1;
    else
        p2=1;
    end
end                                             %p has the index of the begining of the silent part

while p3==0                                     %looks for the second pilot after the silent part
if sum(test(p:p+99).*test(p:p+99))>0.02          %threshold is 0.2 for computer 10
        p3=1;
    else
        p=p+1;
    end
end

p1=p;
```

```
while p4==0                                     %looks for the second pilot after the silent part
if sum(test(p1:p1+99).*test(p1:p1+99))>0.0001          %threshold is 0.2 for computer 10
        p1=p1+1;
    else
        p4=1;
    end
end
test5=test(100:p-1);                            %from begining first pilot to begining of the second pilot
test6=test(600+p+length(ti):p1+600);               %from second pilot to the end of recording


%%%%%%%%%%%%%%%%%%%%%%%
%%%%First Password%%%%%
%%%%%%%%%%%%%%%%%%%%%%%

%Phase detector for the first password
%Correlation algorithm is applied
%a cosine signals is multiplied and summed with the begining of the pilot
%while the phase shift is swept
%at the value of the maximum sum the signals are in phase

theta1=0;
thera1=0;
k1=0;
phaser1=cos(2*pi*(fc)*ti+theta1);               %sample multiplexer signal1
for theta1 = 0:1/fc:2*pi                        %theta1 is swept from 0 to 2pi
ff1=test5(length(phaser1):2*length(phaser1)-1)'.*cos(2*pi*(fc)*ti+theta1);
%the first bits of data are chopped since they are distorted due to initial transients
    if(sum(ff1)>k1)                             %if the correlation is maximum upto now
        k1=sum(ff1);                            %updates the phase thera1
        thera1=theta1;
    end
end
t1=[0:1/fs:(length(test5)-length(phaser1))*(1/fs)];
%time vector length of the signal of first password
mul1a= 2^(1/2)*cos(2*pi*(fc)*t1+thera1);         %multiplexer signal for first brabch of the first password
mul1b=-2^(1/2)*sin(2*pi*(fc)*t1+thera1);         %multiplexer signal for second brabch of the first password
a1a=test5(length(phaser1):end)'.*mul1a;          %carrier cos signal removed
a1b=test5(length(phaser1):end)'.*mul1b;          %carrier -sin signal removed
% figure
% plot(a1a)
% figure
% pwelch(a1a);                                  %signal around 0 and 2fc
% figure
% plot(a1b)
% figure
% pwelch(a1b);                                  %signal around 0 and 2fc



flp =  400 / (fs/2);                            %low-pass filter with flat response till 180Hz
[p,z] = butter(8, flp);                         %8th order butterworth
b1a = filtfilt(p, z, a1a);                      %no phase filter applied to first branch
b1b = filtfilt(p, z, a1b);                      %no phase filter applied to second branch
% figure                                        %
% freqz(p,z,128,fs)                             %freq. response of filter



% figure                                        %plot the signal after the filter
% plot(b1a)
% figure
% pwelch(b1a)                                   %2fc and noise removed
% figure                                        %plot the signal after the filter
% plot(b1b)
% figure
% pwelch(b1b)                                   %2fc and noise removed





RS=45;                                          %Rb=90 so Rs=Rb/2
N=fs/RS;                                        %number of samples per symbol
delay1=0;                                       %used for syn
delay_No1=5;                                    %means how many periods need to memoried for delay block,
memory_block1=zeros(1,2*delay_No1);             %the more No., the more acurancy of result.

Ht1=ones(1,N);                                  %it is impulse response of match filter
```

```
e1a=b1a(30:end);                                %branch of cosine
e1b=b1b(30:end);                                %branch of sine
% Convolution
J1a=conv(e1a,Ht1);
J1b=conv(e1b,Ht1);


% sampling synchronization
group11=zeros(1,round(length(J1a)/N)-1);
lg1=[0:length(group11)-1]*N;
max1=0;
for i1=1:N                                      %using a series of impulses which has constant distance between
each impulse, Ts.
    group11=J1a(lg1+i1);
    if sum(abs(group11)) > max1                 %finding the maximum value of train
        delay1=i1;
        max1=sum(abs(group11));
    end

end


% sampling after match filter
Input_No1=length(b1a)/N;

for i1=1:Input_No1                               %start from the first sampling instant, and take every next Nth
sample in the future
   JJ1a=J1a(delay1:end);
   JJ1b=J1b(delay1:end);
   K1a(i1)=J1a(N*(i1)+delay1);
   K1b(i1)=J1b(N*(i1)+delay1);


   if K1a(i1)>1 && K1b(i1)>1                     %translate sampling value to binary string
      L1(2*i1-1:2*i1)=[1,1];
   elseif K1a(i1)<-1 && K1b(i1)>1
      L1(2*i1-1:2*i1)=[0,1];
   elseif K1a(i1)>1 && K1b(i1)<-1
      L1(2*i1-1:2*i1)=[1,0];
   else
      L1(2*i1-1:2*i1)=[0,0];
   end
end
% plot(L1,'.r')
% figure
% plot(K1a,'.r')
% hold on
% plot(K1b,'.b')
% figure
% plot(J1a,'.r')
% hold on
% plot(J1b,'.b')




binary_array1=L1';
%-------search for the first "\"---------------------
m1=1;
n1=1;
while m1
    binary_array_sample1=binary_array1(n1:n1+6);         %compare groups of 7 bits with the "\" ASCII code
    diff1=binary_array_sample1-[1 0 1 1 1 0 0]';         %when it is found, go to processing
    if diff1==[0 0 0 0 0 0 0]'
        m1=0;
    end
    n1=n1+1;                                              %denotes the position of the "\" within the array
end
%-----------------save ASCII to ascii_array-------------------
na=n1;                                                    %first backslash
%---------------
m1=1;
n1=n1-1;
i1=1;
while m1
   diff1=binary_array1(n1+7:n1+13)-[1 0 1 1 1 0 0]';       %compare groups of 7 bits with the "\" ASCII code
   if diff1==[0 0 0 0 0 0 0]'                              %when it is found, go to conversion
       m1=0;
       break
   end
   ascii_array1(:,i1)=binary_array1(n1+7:n1+13);           %saves the data till the second "\" is found
   n1=n1+7;
   i1=i1+1;
end
%--------test--compare input with output----

aaaa1=bi2de(ascii_array1','left-msb');                     %conversion from binary to decimal values
password1=char(aaaa1')                                     %conversion to characters from binary data stream

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%Second Password%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

%Phase detector for the first password
%Correlation algorithm is applied
%a cosine signals is multiplied and summed with the begining of the pilot
%while the phase shift is swept
%at the value of the maximum sum the signals are in phase

theta2=0;
thera2=0;
k2=0;
phaser2=cos(2*pi*(fc)*ti+theta2);               %sample multiplexer signal2
for theta2 = 0:1/fc:2*pi                        %theta2 is swept from 0 to 2pi
ff2=test6(3*length(phaser2):4*length(phaser2)-1)'.*cos(2*pi*(fc)*ti+theta2);
%the first bits of data are chopped since they are distorted due to power
%detector alogrithm
    if(sum(ff2)>k2)
        k2=sum(ff2);                            %if the correlation is maximum upto now
        thera2=theta2;                          %updates the phase thera1
    end
end
t2=[0:1/fs:(length(test6)-3*length(phaser2))*(1/fs)];
%time vector length of the signal of second password
mul2a=2^(1/2)*cos(2*pi*(fc)*t2+thera2);         %multiplexer signal for first branch of the second password
mul2b=-2^(1/2)*sin(2*pi*(fc)*t2+thera2);        %multiplexer signal for first branch of the second password

a2a=test6(3*length(phaser2):end)'.*mul2a;       %carrier cos signal removed
a2b=test6(3*length(phaser2):end)'.*mul2b;       %carrier -sin signal removed

% figure
% plot(a2a)
% figure
% pwelch(a2a);                                  %signal around 0 and 2fc
% figure
% plot(a2b)
% figure
% pwelch(a2b);                                  %signal around 0 and 2fc



flp =  400 / (fs/2);                            %low-pass filter with flat response till 180Hz
[p,z] = butter(8, flp);                         %8th order butterworth
b2a = filtfilt(p, z, a2a);                      %no phase filter applied
b2b = filtfilt(p, z, a2b);                      %no phase filter applied
% figure                                        %
% freqz(p,z,128,fs)                             %freq. responce of filter



% figure                                        %plot the signal after the filter
% plot(b2a)
% figure
% pwelch(b2a)                                   %2fc and noise removed
% figure                                        %plot the signal after the filter
% plot(b2b)
% figure
% pwelch(b2b)                                   %2fc and noise removed



SR=45;                                          %Rb=90 so Rs=Rb/2
N=fs/SR;                                        %number of samples per symbol
delay2=0;                                       %used for syn
delay_No2=5;                                    %means how many periods need to memoried for delay block,
memory_block2=zeros(1,2*delay_No2);             %the more No., the more acurancy of result.

Ht2=ones(1,N);                                  %it is impulse response of match filter
```

```
e2a=b2a(30:end);                                %branch of cosine
e2b=b2b(30:end);                                %branch of sine
% Convolution
J2a=conv(e2a,Ht2);
J2b=conv(e2b,Ht2);




group12=zeros(1,round(length(J2a)/N)-1);
lg2=[0:length(group12)-1]*N;
max2=0;
for i2=1:N                                      %using a series of impulses which has constant distance between
each impulse, Ts.
    group12=J2a(lg2+i2);
    if sum(abs(group12)) > max2                 %finding the maximum value of train
        delay2=i2;
        max2=sum(abs(group12));
    end

end

% sampling after match filter
Input_No2=length(b2a)/N;
for i2=1:Input_No2                              %start from the first sampling instant, and take every next Nth
sample in the future
   JJ2a=J2a(delay2:end);
   JJ2b=J2b(delay2:end);
   K2a(i2)=J2a(N*(i2)+delay2);
   K2b(i2)=J2b(N*(i2)+delay2);


   if K2a(i2)>1 && K2b(i2)>1                    %translate sampling value to binary string
      L2(2*i2-1:2*i2)=[1,1];
   elseif K2a(i2)<-1 && K2b(i2)>1
      L2(2*i2-1:2*i2)=[0,1];
   elseif K2a(i2)>1 && K2b(i2)<-1
      L2(2*i2-1:2*i2)=[1,0];
   else
      L2(2*i2-1:2*i2)=[0,0];
   end
end
% plot(K2a,'.r')
% hold on
% plot(K2b,'.b')

binary_array2=L2';
%-------search for the first "\"---------------------
m2=1;
n2=1;
while m2
    binary_array_sample2=binary_array2(n2:n2+6);             %compare groups of 7 bits with the "\" ASCII code
    diff2=binary_array_sample2-[1 0 1 1 1 0 0]';             %when it is found, go to processing
    if diff2==[0 0 0 0 0 0 0]'
        m2=0;
    end
    n2=n2+1;                                                  %denotes the position of the "\" within the array
end
%-----------------save ASCII to ascii_array-------------------

nb=n2;                                                        %first backslash
%---------------
m2=1;
n2=n2-1;
i2=1;
while m2
   diff2=binary_array2(n2+7:n2+13)-[1 0 1 1 1 0 0]';          %compare groups of 7 bits with the "\" ASCII code
   if diff2==[0 0 0 0 0 0 0]'                                 %when it is found, go to conversion
       m2=0;
       break
   end
   ascii_array2(:,i2)=binary_array2(n2+7:n2+13);              %saves the data till the second "\" is found
   n2=n2+7;
   i2=i2+1;
end
%--------test--compare input with output----

aaaa2=bi2de(ascii_array2','left-msb');                       %conversion from binary to decimal values
password2=char(aaaa2')                                       %conversion to characters from binary data stream
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%plots%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
if(showplots==1)
test1=wavrecord(31*fs,fs,1);                    %record the the detected input
figure
plot(t/fs,test)                                 %input is plotted in time domain
xlabel('time[s]')
ylabel('amplitude of the recorded signal')
title ('input')

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%constellation plots%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%First Part

if(mod(na,2)==1)                            %na shows the index of first "\" in the combined bitstream
    na=na+1;                                %for the individual branches need to be divided by two
end                                         %so need to be divisible by two
if(mod(n1,2)==1)                            %n1 shows the index of second "\" in the combined bitstream
    n1=n1-1;
end

a_max1a=max(abs(K1a(na/2:n1/2)));           %data without information are removed
PP1a=K1a/a_max1a;
a1a=length(K1a(na/2:n1/2));
KK1a=abs(K1a(na/2:n1/2));
abs_sum1a=sum(KK1a);
KKmax1a=max(KK1a);
x1a=[0:1:KKmax1a];
aa1a=hist(KK1a,x1a);
bb1a=max(aa1a(10:length(aa1a))); %seek the peak value of the histogram except the values around 0
aa_peak1a=find(aa1a==bb1a);
K1asd=K1a(na/2:n1/2)/(max(aa_peak1a)/(2^1/2));%power normalization

a_max1b=max(abs(K1b(na/2:n1/2)));           %data without information are removed
PP1b=K1b/a_max1b;
a1b=length(K1b(na/2:n1/2));
KK1b=abs(K1b(na/2:n1/2));
abs_sum1b=sum(KK1b);
KKmax1b=max(KK1b);
x1b=[0:1:KKmax1b];
aa1b=hist(KK1b,x1b);
bb1b=max(aa1b(10:length(aa1b))); %seek the peak value of the histogram except the values around 0
aa_peak1b=find(aa1a==bb1a);
K1bsd=K1b(na/2:n1/2)/(max(aa_peak1b)/(2^1/2));%power normalization


figure
hold on
j=0;
for i=1:length(K1asd)
    plot(K1asd,K1bsd,'.r')
    title('constellation for the first password')
end

%Second Part

if(mod(nb,2)==1)                            %nb shows the index of first "\" in the combined bitstream
    nb=nb+1;                                %for the individual branches need to be divided by two
end                                         %so need to be divisible by two
if(mod(n2,2)==1)                            %n1 shows the index of second "\" in the combined bitstream
    n2=n2-1;
end

a_max2a=max(abs(K2a(nb/2:n2/2)));           %data without information are removed first branch
PP2a=K2a/a_max2a;
a2a=length(K2a(nb/2:n2/2));
KK2a=abs(K2a(nb/2:n2/2));
abs_sum2a=sum(KK2a);
KKmax2a=max(KK2a);
x2a=[0:1:KKmax2a];
aa2a=hist(KK2a,x2a);
bb2a=max(aa2a(10:length(aa2a))); %seek the peak value of the histogram except the values around 0
aa_peak2a=find(aa2a==bb2a);
K2asd=K2a(nb/2:n2/2)/(max(aa_peak2a)/(2^1/2));%power normalization

a_max2b=max(abs(K2b(nb/2:n2/2)));           %data without information are removed second branch
```

```
PP2b=K2b/a_max2b;
a2b=length(K2b(nb/2:n2/2));
KK2b=abs(K2b(nb/2:n2/2));
abs_sum2b=sum(KK2b);
KKmax2b=max(KK2b);
x2b=[0:1:KKmax2b];
aa2b=hist(KK2b,x2b);
bb2b=max(aa2b(10:length(aa2b))); %seek the peak value of the histogram except the values around 0
aa_peak2b=find(aa2a==bb2a);
K2bsd=K2b(nb/2:n2/2)/(max(aa_peak2b)/(2^1/2));%power normalization


figure
hold on
j=0;
for i=1:length(K2asd)
    plot(K2asd,K2bsd,'*g')
    title('constellation for the second password')
end

%%%%%%%%%%%%%%%%%%%%
%%%%eye diagrams%%%%
%%%%%%%%%%%%%%%%%%%%
%First Password
%First Branch (cos)
eyediagram(JJ1a(na*600:n1*600),1200)            %data without information are removed
title('eye diagram for cosine branch of the first password')
xlabel('time [Ts]')
ylabel('amplitude')

%Second Branch (-sin)
eyediagram(JJ1b(na*600:n1*600),1200)            %data without information are removed
title('eye diagram for sine branch of the first password')
xlabel('time [Ts]')
ylabel('amplitude')


%Second Password
%First Branch (cos)
eyediagram(JJ2a(nb*600:n2*600),1200)            %data without information are removed
title('eye diagram for cosine branch of the second password')
xlabel('time [Ts]')
ylabel('amplitude')

%Second Branch (-sin)
eyediagram(JJ2b(nb*600:n2*600),1200)            %data without information are removed
title('eye diagram for sine branch of the second password')
xlabel('time [Ts]')
ylabel('amplitude')
end
toc                                             %time spent while decoding
```