```
%BPSK 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 increases
 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 0.05 for comp10
 if powersample>=0.05                           %if the recorded signal is above threshold
     powerin=1;                                 %loop is finished and 'input detected!' message
     display('input detected!')                 %indicating that a sound is detected is displayed
 end                                            %a high pass and band pass filter is applied
end                                             %with passband around 2100Hz
% toc                                             %standby time

% test = wavread('Team11_BPSK_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




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



% Input Chopper
% Divides the input in two parts
% For 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 0.003 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 0.02 for computer 10
        p3=1;
    else
        p=p+1;
    end
end
p1=p;
while p4==0                                      %looks for the silent part after the second password
if sum(test(p1:p1+99).*test(p1:p1+99))>0.0001    %threshold is 0.0001 for computer 10
        p1=p1+1;
```

```
    else
        p4=1;
    end
end
test5=test(1:p-1);                              %from first pilot to begining of the second pilot
test6=test(p+length(ti)+100:p1+600);            %from second pilot to the end of second password

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

%Phase detector for the first password
%Correlation algorithm is applied
%a cosine signal 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
mul1=2^(1/2)*cos(2*pi*(fc)*t1+thera1);          %multiplexer signal for the first password

a1=test5(length(phaser1):end)'.*mul1;           %carrier signal removed
% figure
% plot(a1)
% figure
% pwelch(a1);                                   %signal around 0 and 2fc



flp =  275 / (fs/2);                            %low-pass filter with flat response till 90Hz
[p,z] = butter(8, flp);                         %8th order butterworth
b1 = filtfilt(p, z, a1);                        %no phase filter applied
% figure                                        %
% freqz(p,z,128,fs)                             %freq. responce of filter



% figure                                        %plot the signal after the filter
% plot(b1)
% figure
% pwelch(b1)                                    %2fc signal removed



BR=90;                                          %bitrate is symbol rate and 90

N=fs/BR;                                        %number of samples per bit.
delay1=0;                                       %this will denote the position of the first sampling instant
delay_No1=3;                                    %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
e1=b1(70:end);

% convolution of the filtered signal with a square pulse for matched
% filtering

J1=conv(e1,Ht1);

% sampling synchronization

group1=zeros(1,round(length(J1)/N)-1);
lg1=[0:length(group1)-1]*N;
max1=0;
for i1=1:N                      %using a series of impulses which has constant distance between each impulse,
Ts.
```

```
    group1=J1(lg1+i1);
    if sum(abs(group1)) > max1  %finding the maximum value of train
        delay1=i1;
        max1=sum(abs(group1));
    end

end

% sampling after match filter
Input_No1=length(b1)/N;
for i1=delay_No1:Input_No1      %start from the first sampling instant, and take every next Nth sample in the
future
   JJ1=J1(delay1:end);
   K1(i1)=J1(N*(i1)+delay1);

   if K1(i1)<0                  %translate sampling value to binary string
      L1(i1)=[0];
   else
      L1(i1)=[1];
   end
end

% plot(K1,'.g')
% hold on
% plot(L1,'.r')


binary_array1=L1';
%-------search for the first "\"---------------------
m1=1;
n1=1;
while m1
    binary_array_sample1=binary_array1(n1:n1+6);
    diff1=binary_array_sample1-[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 processing
        m1=0;
    end
    n1=n1+1;                                     %denotes the position of the "\" within the array
end
%-----------------save ASCII to ascii_array-------------------

na=n1;                                           %first backslash
%-------search for the second "\"---------------------
m1=1;
n1=n1-1;
i1=1;
while m1
   diff1=binary_array1(n1+7:n1+13)-[1 0 1 1 1 0 0]';
   if diff1==[0 0 0 0 0 0 0]'
       m1=0;                                        %compare groups of 7 bits with the "\" ASCII code
       break                                        %when it is found, go to conversion
   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

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)                             %if the correlation is maximum upto now
```

```
        k2=sum(ff2);                            %updates the phase thera1
        thera2=theta2;
    end
end
t2=[0:1/fs:(length(test6)-3*length(phaser2))*(1/fs)];
%time vector length of the signal of second password
mul2=2^(1/2)*cos(2*pi*(fc)*t2+thera2);          %multiplexer signal for the second password


a2=test6(3*length(phaser2):end)'.*mul2;         %carrier signal removed
% figure
% plot(a2)
% figure
% pwelch(a2);                                   %signal around 0 and 2fc



flp =  275 / (fs/2);                            %low-pass filter with flat response till 90Hz
[p,z] = butter(8, flp);                         %8th order butterworth
b2 = filtfilt(p, z, a2);                        %no phase filter applied
% figure                                        %
% freqz(p,z,128,fs)                             %freq. responce of filter


% figure                                        %plot the signal after the filter
% plot(b2)
% figure
% pwelch(b2)                                    %2fc signal removed



BR=90;                                          %bitrate is symbol rate and 90
N=fs/BR;                                        %number of samples per bit.
delay2=0;                                       %this will denote the position of the first sampling instant
delay_No2=3;                                    %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
e2=b2(70:end);

% convolution of the filtered signal with a square pulse for matched
% filtering


J2=conv(e2,Ht2);



% sampling synchronization
group2=zeros(1,round(length(J2)/N)-1);
lg2=[0:length(group2)-1]*N;
max2=0;
for i2=1:N                                      %using a series of impulses which has constant distance between
each impulse, Ts.
    group2=J2(lg2+i2);
    if sum(abs(group2)) > max2                  %finding the maximum value of train
        delay2=i2;
        max2=sum(abs(group2));
    end

end

% sampling after match filter
Input_No2=length(b2)/N;
for i2=delay_No2:Input_No2          %start from the first sampling instant, and take every next Nth sample in
the future
   JJ2=J2(delay2:end);
   K2(i2)=J2(N*(i2)+delay2);

   if K2(i2)<0                      %translate sampling value to binary string
      L2(i2)=[0];
   else
      L2(i2)=[1];
   end
end
% plot(K2,'.g')
% hold on
% plot(L2,'.r')
```

```
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

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%plots%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
if(showplots==1)
t=[0:1:length(test)-1];                         %time vector is created
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
a_max=max(abs(K1(na:n1)));                      %data without information are removed
a=length(K1(na:n1));
KK=abs(K1(na:n1));
abs_sum=sum(KK);
KKmax=max(KK);
x=[0:1:KKmax];
aa=hist(KK,x);
bb=max(aa(10:length(aa)));                      %seek the peak value of the histogram, disregard the values
around 0
aa_peak=find(aa==bb);
K1sd=K1(na:n1)/(max(aa_peak)/(2^1/2));          %power normalization
figure
hold on
j=0;
for i=1:length(K1sd)
    plot(K1sd(i),j,'.r')
    title('constellation for the first password')
end

%Second Part
a_max2=max(abs(K2(nb:n2)));                     %data without information are removed
a2=length(K2(nb:n2));
KK2=abs(K2(nb:n2));
abs_sum2=sum(KK2);
KKmax2=max(KK2);
x2=[0:1:KKmax2];
aa2=hist(KK2,x2);
bb2=max(aa2(10:length(aa2)));                   %seek the peak value of the histogram, disregard the values
around 0
```

```
aa_peak2=find(aa2==bb2);
K2sd=K2(nb:n2);
K2sd=K2sd/(max(aa_peak2)/(2^1/2));              %power normalization
figure
 hold on
j2=0;
for i=1:length(K2sd)
    plot(K2sd(i),j2,'*g')
    title('constellation for the second password')
end

%%%%%%%%%%%%%%%%%%%%
%%%%eye diagrams%%%%
%%%%%%%%%%%%%%%%%%%%

%First Part
eyediagram(JJ1(na*600:n1*600),600)              %data without information are removed
title('eye diagram for the first password')
xlabel('time [Ts]')
ylabel('amplitude')

%Second Part
eyediagram(JJ2(nb*600:n2*600),600)              %data without information are removed
xlabel('time [Ts]')
ylabel('amplitude')
title('eye diagram for the second password')
end

toc                                             %time spent while decoding
```