Experiment # 7, 8

To study performance analysis of coherent and non-coherent BPSK demodulation in the presence of noise.

Experiment # 7

Purpose: Analysis of the performance of the coherent demodulation in the presence of noise.

Equipment and materials:

Equipment Required

i) PC with MATLAB and VSA software
ii) ii) LABVIEW software
iii) Arbitrary function generator
iv) Dream Catcher transmitter and receiver trainer kit
v) Signal Analyser
vi) NI down 1convertor
vii) Local oscillator signal generator

Accessories Required

i) SMA(m)-to-SMA(m) coaxial cable
ii) SMA (M) to BNC connector
iii) Antenna

Procedure:

1) Write a MATLAB code to generate use bits and write them into a CSV file.

2) Open the LABVIEW and upload the bits into function generator.

3) Set all setting of function generator for proper transmission of data bits.

4) Select the user bits file and apply other required setting on transmitter and receiver front panels.

5) Run the program to transmit and receive the data via hardware.

6) Collect the received data.

7) Use the original transmitted and extracted user bits to find the bits in error.

MATLAB code

Transmitter code:

```
clc;
clear all;
close all;
N=2^7; % Number of bits
% ~~~~~~~~~~~~~ text to binary ~~~~~~~~~~~~~~~~~~~%
fid = fopen('testingCopy.txt');
    x=fread(fid,'*char');
    binary = dec2bin(x,8);
    binary_t=transpose(binary);
    txt_to_bin=binary_t(:)-'0';
    %%%%%%%%%%%%%%%%%%binery to tax%%%%%%%%%%%%%%%
   % S = reshape(txt_to_bin, length(txt_to_bin)/8, 8)
   %~~~~~~~~~~~~~~~~~~~adding syncronization bits~~~~~~~~~~~~~~~~~
```

```
h_pn=commsrc.pn('GenPoly',[7 6 0],'InitialStates',[0 0 0 0 0 0
1],'NumBitsOut',N);
syncronization_bits=generate(h_pn);
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [sync bits, data bits]~~~~~~~~~~~~~~~~~
    tx_text=[syncronization_bits;txt_to_bin]
 % ~~~~~~~~~~~~BIT Sequence to Signal (Upsampling)~~~~~~~~~~~~~~~~~~~~%
bits_I=2*(tx_text-0.5);
sig_ipt_I=upsample(bits_I,8
%~~~~~~~~~~~~~~~~Rectangular filter~~~~~~~~~~~~~~~~%
ract_fil=ones(8,1);
sig_ract=conv(ract_fil, sig_ipt_I);%
sig_ract_I=sig_ract/sqrt(var(sig_ract));

csvwrite('Coh_Rect_text_ref.csv',tx_text);
csvwrite('Coh_Rect_text_data.csv',sig_ract_I);
```

Receiver Code:

```
clc;
clear all;
close all;
spsym=8;     % samples per symbol
%~~~~~~~~~~~~~~~~~~~~~ Acquireing the data ~~~~~~~~~~~~~~~~~~~~%
RX_ss_tmp = csvread ('acquired_coherent_7_22.csv',0,0);
figure;
plot(RX_ss_tmp(1:2000,:));
RX_ss=complex(RX_ss_tmp(:,1),RX_ss_tmp(:,2));
Ract_filt=ones(spsym,1);              %Filter Length = spsym
XX_signal = conv(Ract_filt,RX_ss);    % Ractangular pulse shaped signal
RX_signal =XX_signal/max([real(XX_signal);imag(XX_signal)]); %normalizing
the convolved signal
%~~~~~~~~~~~~~~~~~~~~~~~~~ EYE Diagram ~~~~~~~~~~~~~~~~~~~~~~~~~~~
eye_len=spsym;
eye_frame_len=floor(length(RX_signal)/eye_len);
I_eye=zeros(eye_len,eye_frame_len);
Q_eye=zeros(eye_len,eye_frame_len);
for i=1:eye_frame_len
    I_eye(:,i)= real(RX_signal((i-1)*eye_len+1:i*eye_len,1));
    Q_eye(:,i)=imag(RX_signal((i-1)*eye_len+1:i*eye_len,1));
end
figure;
plot(I_eye(:,1:100));
%~~~~~~~~~~~~~Sampling Instant Identification ~~~~~~~~~~~~~~~~~~~~~~
eye_var=zeros(eye_len,1);
for i=1:eye_frame_len
    for j=1:eye_len
            eye_var(j,1)=eye_var(j,1)+I_eye(j,i)^2;
    end
end
eye_var=eye_var/eye_frame_len;
[~,eye_offset]=max(eye_var(1:spsym));

%~~~~~~~~ Sample the symbols by downsampling ~~~~~~~~~~~~~~~~~~~~~~
Symbols=zeros(floor(length(RX_signal)/spsym),1);
for i=1:length(Symbols)-1
    Symbols(i,1)=RX_signal((i-1)*spsym+eye_offset,1);
end
figure
plot(Symbols(1:100))
```

```
figure
plot(atan(imag(Symbols(1:1000))./real(Symbols(1:1000))));
%~~implementation of phase and frequency correction(PLL/Costas
loop)~~~~~~~~%
K1_PLL =0.0313;
K2_PLL =2.49e-4;
unwrap_phi_array = zeros(size(Symbols));
phi_array = zeros(size(Symbols));
freq_array = zeros(size(Symbols));
filt_phi_array = zeros(size(Symbols));
filt_freq_array = zeros(size(Symbols));
phi_array(1:2)=atan(imag(Symbols(1:2))./real(Symbols(1:2)));
unwrap_phi_array(1:2)=phi_array(1:2);
for i=3:length(Symbols)
    phi=atan(imag(Symbols(i))/real(Symbols(i)));
    phi1(i)=phi;
    old_phi=phi_array(i-1);
    if (phi-old_phi < -pi/2)
        freq = pi+phi-old_phi;
    elseif (phi-old_phi > pi/2)
        freq = -pi+phi-old_phi;
    else
        freq = phi-old_phi;
    end
    freq_array(i)=freq;
    unwrap_phi_array(i)=unwrap_phi_array(i-1)+freq;
    phi_array(i)=phi;
    % filter cofficient and equation is given in page 411 and 412 of given
    % google drive link
    %
https://drive.google.com/file/d/1FpZeWW0sb8Ve_C0QDALye7r1kmmb3nmd/view?usp=
sharing

    filt_phi_array(i) = (2-K1_PLL-K2_PLL)*filt_phi_array(i-1) ...
                        -(1-K1_PLL)*filt_phi_array(i-2) ...
                        +(K1_PLL+K2_PLL)*unwrap_phi_array(i-1) ...
                        -(K1_PLL)*unwrap_phi_array(i-2);
    Symbols(i) = Symbols(i)*complex(cos(filt_phi_array(i)),-
sin(filt_phi_array(i)));

    end

figure
plot(Symbols(1:100));
%~~~~~~~~~~~~~~~~~~Loading transmitted data for BER
Calulation~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
tx_bits = csvread ('Coh_Rect_text_ref.csv',1);
tx_bits_bipolar=2*tx_bits-1;
%~~~~~~~~~~Finding the starting bit and detect the bits (Works in good
SNR)~~~~~~~~~~~~~~~~~~~~~~~~%
bits_rec_bipolar=sign(real(Symbols));
corval=zeros(length(tx_bits_bipolar),1);
%~~~~~~~~~~~~~~~ Using syncronization bit finding stating start of the data
packet~~~~~~~~~~~~~~~%
for i=1:length(tx_bits)*2
    corval(i,1)=sum(tx_bits_bipolar(1:128).*bits_rec_bipolar(i:i+127));
end
figure;
plot(corval)
```

```
[peak,start_bit_ind]=max(abs(corval)); %searching maximum correlation value
and its index number
start_bit_ind
peak
ss=sign(corval(start_bit_ind));       % required to correct the polarity of
PLL output
bits_rec=(ss*bits_rec_bipolar+1)/2;   %converting 1--->1 and -1---->0
detected_bits=bits_rec(start_bit_ind:start_bit_ind+length(tx_bits)-1);
%chooping out of one packet data
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%

Nof_err_bits=sum(abs(tx_bits-detected_bits)); %finding number of bits get
fliped
BER=Nof_err_bits/length(tx_bits); % Bit error rate
Nof_err_bits
BER

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~binary to text convertion
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
bin_to_text=[0;detected_bits];
bin_to_text=bin_to_text(129:end);
btxt   = reshape(bin_to_text,[8, length(bin_to_text)/8])';
if length(class(btxt))== 6
    text   = char(bin2dec(char(btxt+48)))';
else
    text   = char(bin2dec(btxt))'
end
text
```

Experiment # 8

Purpose: Analysis of the performance of the non-coherent demodulation in the presence of noise.

Equipment and materials:
Equipment Required
i) PC with MATLAB and VSA software

Procedure:

1) Add a MATLAB code for differential encoding and decoding in MATLAB function written in The code for differential encoding and decoding is given as follows:

```
% ~~~~~~~~~Differential Encoding ~~~~~~~~~~~~~~~~~~~~~~~%
diff_bits=zeros(length(tx_text),1);
diff_bits(1)=xor(0,tx_text(1));
for i=2:length(diff_bits)
    diff_bits(i)=xor(diff_bits(i-1),tx_text(i)); % XOR ing privious bit to
current bit
end


%~~~~~~~~~Diferential Decoding~~~~~~~~~~~~~~~~~%
angle_array=[0;atan2(imag(Symbols),real(Symbols))];
angle_diff=abs(angle_array(2:end)-angle_array(1:end-1));
for i=1:length(angle_diff)
    if angle_diff(i)>3*pi/2
        angle_diff(i)=0;
    end
end
bits_rec_bipolar=sign(angle_diff-pi/2);
```