**EXPERIMENT - 5**

**BPSK Modulator and Demodulator over AWGN Channel [1]:**

***Example***

%clear; %clear all stored variables

N=100; %number of data bits

noiseVariance = 0.5; %Noise variance of AWGN channel

data=randn(1,N)>=0; %Generate uniformly distributed random data

Rb=1e3; %bit rate

amplitude=1; % Amplitude of NRZ data

[time,nrzData,Fs]=NRZ\_Encoder(data,Rb,amplitude,'Polar');

Tb=1/Rb;

subplot(4,2,1);

stem(data);

xlabel('Samples');

ylabel('Amplitude');

title('Input Binary Data');

axis([0,N,-0.5,1.5]);

subplot(4,2,3);

plotHandle=plot(time,nrzData);

xlabel('Time');

ylabel('Amplitude');

title('Polar NRZ encoded data');

set(plotHandle,'LineWidth',2.5);

maxTime=max(time);

maxAmp=max(nrzData);

minAmp=min(nrzData);

axis([0,maxTime,minAmp-1,maxAmp+1]);

grid on;

Fc=2\*Rb;

osc = sin(2\*pi\*Fc\*time);

%BPSK modulation

bpskModulated = nrzData.\*osc;

subplot(4,2,5);

plot(time,bpskModulated);

xlabel('Time');

ylabel('Amplitude');

title('BPSK Modulated Data');

maxTime=max(time);

maxAmp=max(nrzData);

minAmp=min(nrzData);

axis([0,maxTime,minAmp-1,maxAmp+1]);

%plotting the PSD of BPSK modulated data

subplot(4,2,7);

h=spectrum.welch; %Welch spectrum estimator

Hpsd = psd(h,bpskModulated,'Fs',Fs);

plot(Hpsd);

title('PSD of BPSK modulated Data');

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

%Adding Channel Noise

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

noise = sqrt(noiseVariance)\*randn(1,length(bpskModulated));

received = bpskModulated + noise;

subplot(4,2,2);

plot(time,received);

xlabel('Time');

ylabel('Amplitude');

title('BPSK Modulated Data with AWGN noise');

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

%BPSK Receiver

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

%Multiplying the received signal with reference Oscillator

v = received.\*osc;

%Integrator

integrationBase = 0:1/Fs:Tb-1/Fs;

for i = 0:(length(v)/(Tb\*Fs))-1,

y(i+1)=trapz(integrationBase,v(int32(i\*Tb\*Fs+1):int32((i+1)\*Tb\*Fs)));

end

%Threshold Comparator

estimatedBits=(y>=0);

subplot(4,2,4);

stem(estimatedBits);

xlabel('Samples');

ylabel('Amplitude');

title('Estimated Binary Data');

axis([0,N,-0.5,1.5]);

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

%Bit Error rate Calculation

BER = sum(xor(data,estimatedBits))/length(data);

%Constellation Mapper at Transmitter side

subplot(4,2,6);

Q = zeros(1,length(nrzData)); %No Quadrature Component for BPSK

stem(nrzData,Q);

xlabel('Inphase Component');

ylabel('Quadrature Phase component');

title('BPSK Constellation at Transmitter');

axis([-1.5,1.5,-1,1]);

%constellation Mapper at receiver side

subplot(4,2,8);

Q = zeros(1,length(y)); %No Quadrature Component for BPSK

stem(y/max(y),Q);

xlabel('Inphase Component');

ylabel('Quadrature Phase component');

title(['BPSK Constellation at Receiver when AWGN Noise Variance =',num2str(noiseVariance)]);

axis([-1.5,1.5,-1,1]);

function [time,output,Fs]=NRZ\_Encoder(input,Rb,amplitude,style)

Fs=16\*Rb; %Sampling frequency ,

oversamplingfactor= 32;

Ts=1/Fs; % Sampling Period

Tb=1/Rb; % Bit period

output=[];

switch lower(style)

case {'manchester'}

for count=1:length(input)

for tempTime=0:Ts:Tb/2-Ts

output=[output (-1)^(input(count))\*amplitude];

end

for tempTime=Tb/2:Ts:Tb-Ts

output=[output (-1)^(input(count)+1)\*amplitude];

end

end

case {'unipolar'}

for count=1:length(input)

for tempTime=0:Ts:Tb-Ts

output=[output input(count)\*amplitude];

end

end

case {'polar'}

for count=1:length(input)

for tempTime=0:Ts:Tb-Ts

output=[output amplitude\*(-1)^(1+input(count))];

end

end

otherwise

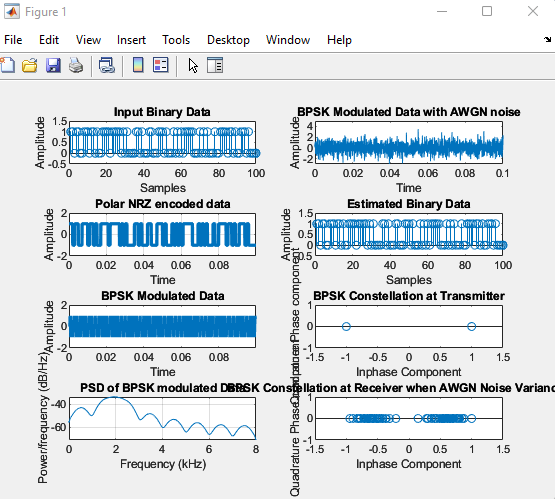
disp('NRZ\_Encoder(input,Rb,amplitude,style)-Unknown method given as ''style'' argument');

disp('Accepted Styles are ''Manchester'', ''Unipolar'' and ''Polar''');

end

time=0:Ts:Tb\*length(input)-Ts;

end



***Exercises***

***1. Design a QPSK Modulator and Demodulator system and evaluate the BER over an AWGN Channel.***

%clear; %clear all stored variables

N=100; %number of data bits

noiseVariance = 0.5; %Noise variance of AWGN channel

data=randn(1,N)>=0; %Generate uniformly distributed random data

Rb=1e3; %bit rate

amplitude=1; % Amplitude of NRZ data

[time,nrzData,Fs]=NRZ\_Encoder(data,Rb,amplitude,'Polar');

Tb=1/Rb;

subplot(4,2,1);

stem(data);

xlabel('Samples');

ylabel('Amplitude');

title('Input Binary Data');

axis([0,N,-0.5,1.5]);

subplot(4,2,3);

plotHandle=plot(time,nrzData);

xlabel('Time');

ylabel('Amplitude');

title('Polar NRZ encoded data');

set(plotHandle,'LineWidth',2.5);

maxTime=max(time);

maxAmp=max(nrzData);

minAmp=min(nrzData);

axis([0,maxTime,minAmp-1,maxAmp+1]);

grid on;

Fc=2\*Rb;

osc1 = sin(2\*pi\*Fc\*time);

osc2= cos(2\*pi\*Fc\*time);

%BPSK modulation

qpskModulated1 = nrzData.\*osc1;

qpskModulated2 = nrzData.\*osc2;

subplot(4,2,5);

plot(time,(qpskModulated1+qpskModulated2));

xlabel('Time');

ylabel('Amplitude');

title('QPSK Modulated Data');

% subplot(4,2,7);

% plot(time,qpskModulated2);

% xlabel('Time');

% ylabel('Amplitude');

% title('QPSK Modulated Data2');

maxTime=max(time);

maxAmp=max(nrzData);

minAmp=min(nrzData);

axis([0,maxTime,minAmp-1,maxAmp+1]);

%plotting the PSD of BPSK modulated data

subplot(4,2,7);

h=spectrum.welch; %Welch spectrum estimator

Hpsd = psd(h,(qpskModulated1+qpskModulated2),'Fs',Fs);

plot(Hpsd);

title('PSD of QPSK modulated Data');

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

%Adding Channel Noise

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

noise1 = sqrt(noiseVariance)\*randn(1,length(qpskModulated1));

noise2 = sqrt(noiseVariance)\*randn(1,length(qpskModulated2));

received1 = qpskModulated1 + noise1;

received2 = qpskModulated2 + noise2;

subplot(4,2,2);

plot(time,(received1+received2));

xlabel('Time');

ylabel('Amplitude');

title('QPSK Modulated Data with AWGN noise');

%BPSK Receiver

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

%Multiplying the received signal with reference Oscillator

v1 = received1.\*osc1;

v2 = received1.\*osc2;

%Integrator

integrationBase = 0:1/Fs:Tb-1/Fs;

for i = 0:(length(v1)/(Tb\*Fs))-1

y(i+1)=trapz(integrationBase,v1(int32(i\*Tb\*Fs+1):int32((i+1)\*Tb\*Fs)));

end

%Threshold Comparator

estimatedBits=(y>=0);

subplot(4,2,4);

stem(estimatedBits);

xlabel('Samples');

ylabel('Amplitude');

title('Estimated Binary Data');

axis([0,N,-0.5,1.5]);

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

%Bit Error rate Calculation

BER = sum(xor(data,estimatedBits))/length(data);

%Constellation Mapper at Transmitter side

subplot(4,2,6);

Q1 = zeros(1,length(nrzData)); %No Quadrature Component for BPSK

Q2 = ones(1,length(nrzData));

scatter(nrzData,Q1);

hold on;

%figure(2);

subplot(4,2,6);

scatter(nrzData,Q2);

xlabel('Inphase Component');

ylabel('Quadrature Phase component');

title('QPSK Constellation at Transmitter');

axis([-1.5,1.5,-1,1]);

%constellation Mapper at receiver side

subplot(4,2,8);

Q1 = zeros(1,length(y)); %No Quadrature Component for BPSK

Q2 = ones(1,length(y));

stem(y/max(y),Q1);

hold on;

%figure(3);

subplot(4,2,8);

scatter(y/max(y),Q2);

%stem(y/max(y),Q);

xlabel('Inphase Component');

ylabel('Quadrature Phase component');

title(['QPSK Constellation at Receiver when AWGN Noise Variance =',num2str(noiseVariance)]);

axis([-1.5,1.5,-1,1]);

function [time,output,Fs]=NRZ\_Encoder(input,Rb,amplitude,style)

Fs=16\*Rb; %Sampling frequency

oversamplingfactor = 32;

Ts=1/Fs; % Sampling Period

Tb=1/Rb; % Bit period

output=[];

switch lower(style)

case {'manchester'}

for count=1:length(input)

for tempTime=0:Ts:Tb/2-Ts

output=[output (-1)^(input(count))\*amplitude];

end

for tempTime=Tb/2:Ts:Tb-Ts

output=[output (-1)^(input(count)+1)\*amplitude];

end

end

case {'unipolar'}

for count=1:length(input)

for tempTime=0:Ts:Tb-Ts

output=[output input(count)\*amplitude];

end

end

case {'polar'}

for count=1:length(input)

for tempTime=0:Ts:Tb-Ts

output=[output amplitude\*(-1)^(1+input(count))];

end

end

otherwise

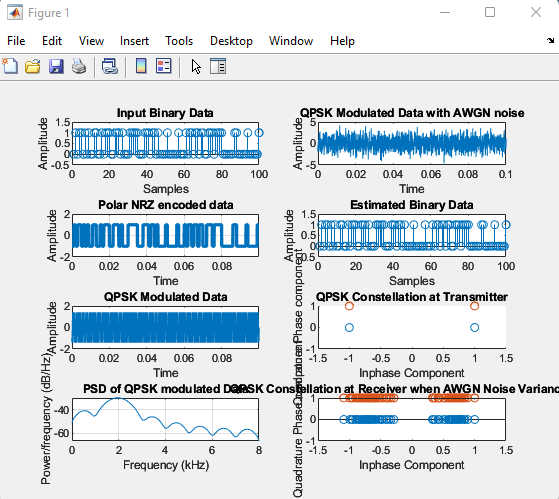
disp('NRZ\_Encoder(input,Rb,amplitude,style)-Unknown method given as ''style'' argument');

disp('Accepted Styles are ''Manchester'', ''Unipolar'' and ''Polar''');

end

time=0:Ts:Tb\*length(input)-Ts;

end

******

***2. Design a FSK Modulator and Demodulator system and evaluate the BER over an AWGN Channel***

clc

close all

clear all

fc1=input('Enter the freq of 1st Sine Wave carrier:');

fc2=input('Enter the freq of 2nd Sine Wave carrier:');

fp=input('Enter the freq of Periodic Binary pulse (Message):');

amp=input('Enter the amplitude (For Both Carrier & Binary Pulse Message):');

amp=amp/2;

t=0:0.001:1;

c1=amp.\*sin(2\*pi\*fc1\*t);

c2=amp.\*sin(2\*pi\*fc2\*t);

subplot(5,1,1);

plot(t,c1)

xlabel('Time')

ylabel('Amplitude')

title('Carrier 1 Wave')

subplot(5,1,2)

plot(t,c2)

xlabel('Time')

ylabel('Amplitude')

title('Carrier 2 Wave')

m=amp.\*square(2\*pi\*fp\*t)+amp;

subplot(5,1,3)

plot(t,m)

xlabel('Time')

ylabel('Amplitude')

title('Binary Message Pulses')

for i=0:1000

if m(i+1)==0

mm(i+1)=c2(i+1);

else

mm(i+1)=c1(i+1);

end

end

subplot(5,1,4)

plot(t,mm)

xlabel('Time')

ylabel('Amplitude')

title('Modulated Wave')

mmawgn = awgn(mm, 30);

subplot(5,1,5),

plot(mmawgn);

title('FSK WAVEFORM WITH NOISE')

