EE352 – Communication Systems II

Final Report

P13. End to end simulation of a Direct Sequence Spread Spectrum (DSSS)

Group Members:

Name: Majid Ahmadov **Number: 210206054**

Name: Mahmut Aksakallı **Number: 210206034**

Name: Alkım Dikmen **Number: 230206056**

A. Introduction

The aim of our project is to make end to end simulation of Direct Sequence Spread Spectrum (DSSS) technique. It's a technique that message signal is multiplied by a pseudo-random sequence to spread message bandwidth so energy of message signal expand a wider spectrum, and it appears as noise. The motivation of DSSS technique comes from channel-capacity theorem. It says that we can have good communication performance by increasing bandwidth even when signal-to-noise power is low. [1] DSSS increases transmit signal bandwidth to remove intersymbol interference (ISI) and narrowband interference. It provides security of transmission because receiver must know pseudo-random sequence to get transmitted data signal. Otherwise, receiver can't detect original message signal, and it only sees transmitted signal as a noise. This property of DSSS technique make it suitable to use for military communication systems. DSSS also enables to usage of a bandwidth by multiple users because each user multiply its message signal by different pseudo-random sequence, and a receiver gets its message signal if it has transmitter pseudo-random sequence. The process of DSSS is shown below.

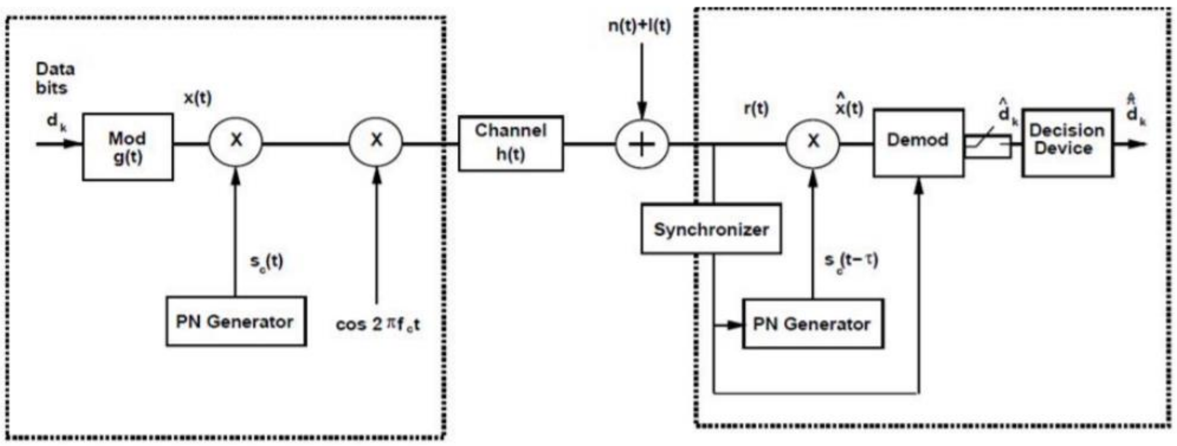

Figure 1: *Block diagram of Spread Spectrum System* [2]

B. Experimental Work

In our project, end to end simulation of DSSS modulation started with generation of message signal at transmitter. We generated 8 bits message signal for better understanding of given figures below. Each bit has 0.1 seconds bit duration. Then, we generated pseudo-random code that has 4 times higher frequency than message signal. This means that each message bit coded by 4 randomly generated bit at DSSS signal. To represent binary data, we used non-return-to-zero (NRZ) line code scheme because we chose BPSK modulation to transmit our signal. BPSK modulation has 180 degree phase shift between carriers of binary 0 and 1 so binary-1 represented as 1 while binary-0 represented as -1. We obtained DSSS signal with multiplying message signal and pseudo-random code. The receiver must be aware of this pseudo-random code. Otherwise, it can't get transmitted message signal, and transmitted signal appears as a noise at receiver. Figure 2 shows generated 8 bits message signal, pseudo-random code and modulated DSSS signal.

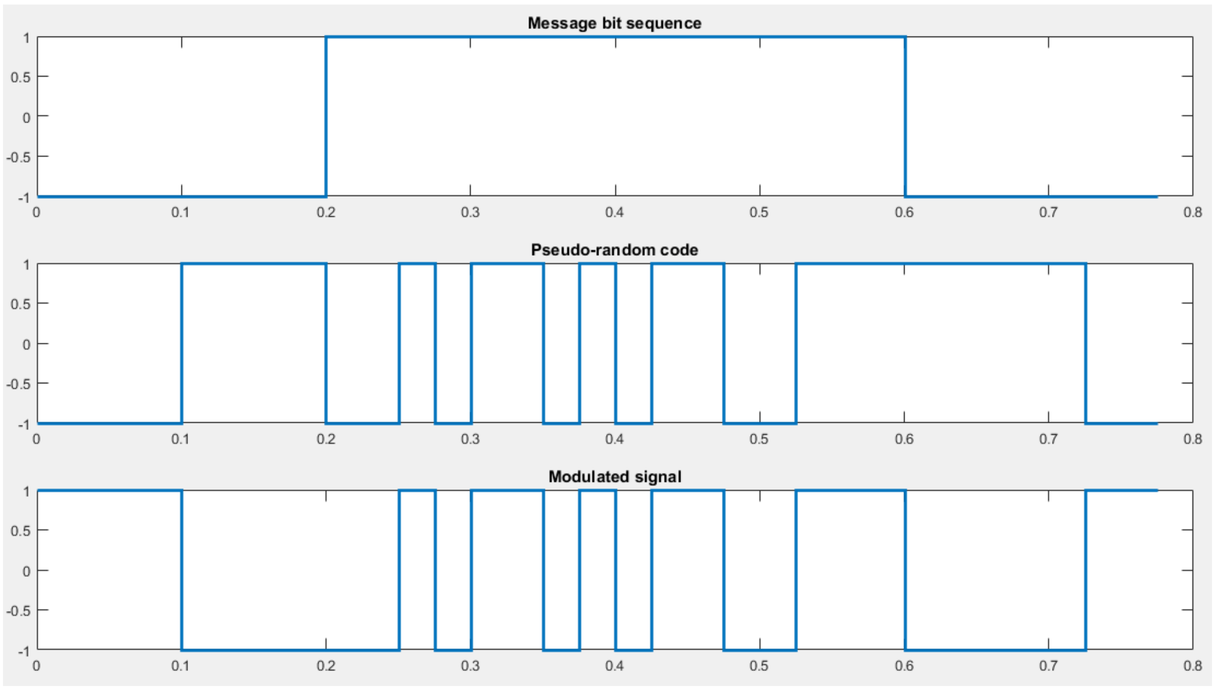

Figure 2: *Message, pseudo-random code and DSSS modulated signal*

Channel-capacity theorem says that increasing bandwidth improves error performance of communication even if SNR is low. DSSS modulation uses advantage of this theorem so message spectrum spread over higher bandwidth while message power decreases noise level. We can observe effects of DSSS modulation at frequency spectrum. Figure 3 shows frequency spectrum of 8 bits message signal, pseudo-random code and modulated DSSS signal. We observed that bandwidth of message expanded, and power of message spread over higher bandwidth than original message signal.

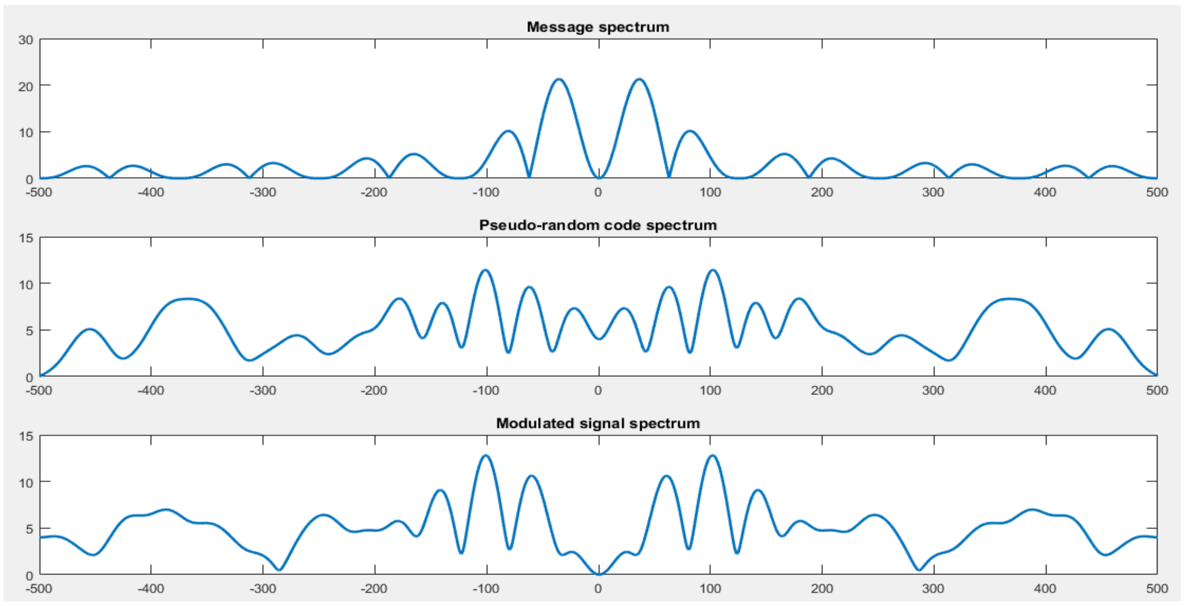

Figure 3: *Frequency Spectrums*

Final step of DSSS modulation at transmitter is multiplying DSSS signal with carrier of BPSK. We handled phase difference between carriers of binary-0 and bnary-1 using non-return-to-zero coding so we just multiplied DSSS signal with a carrier that has 100 Hz frequency. We analyzed effects of noise in performance evaluation step so we didn’t add any noise at channel. At receiver, we multiplied received signal with pseudo-random code. Then, we multiplied encoded signal with carrier to demodulate signal, and send it to decision device. The critical issue at this point is that the receiver must have same pseudo-random code with transmitter otherwise, it can’t reach original message. This feature of DSSS modulation makes it suitable for secure communication. Non-authorized listeners can’t decode message because they don’t have pseudo-random code. Last plot at Figure 4 shows received message when receiver uses different pseudo-random code than transmitter. It’s obvious that receiver can’t decode original message.

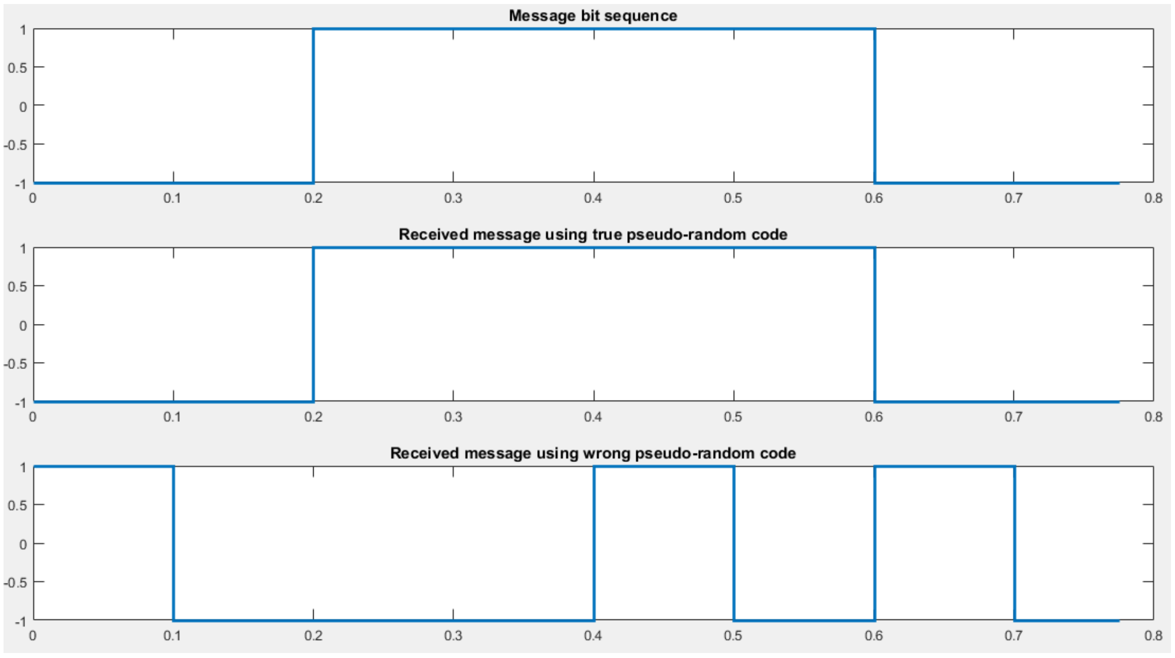

Figure 4: *Original Message and results with true and wrong key at receiver*

Decision device integrates signal over a bit duration intervals, and determine binary value of received signal using a threshold value between binary-0 and binary-1. The result of decision device is shown above in Figure 4.

C. Results

After we obtain the message signal at receiver with zero error rate, we first tested DSSS modulation with Additive White Gaussian Noise. Results are obtained in terms of Bit Error Rate (BER). BER is nothing but a ratio of number of error bits and total number of bits transmitted during a specific period. While SNR value decreases, we checked BER and analyzed. We expected to get good BER because DSSS modulation is known for its higher resistance to noise. We repeated each calculation 1000 times and got average value. As a result, we obtained Monte Carlo analyses of BER with different SNR values. Figure5 shows BER vs SNR plot while message signal has length of 100 bits.

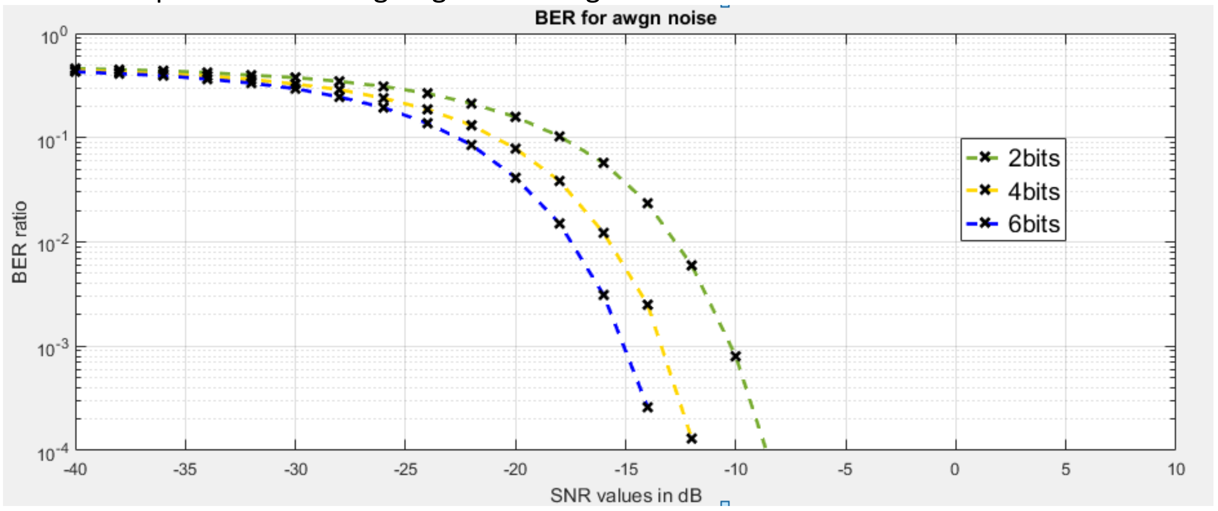

Figure 5: *Additive White Gaussian Noise performance*

It is easy to see that the more pseudo-number bits using, the better BER results obtaining. While SNR value is under -30 dB, BER approaches its max value 0.5. Theoretical values are marked with “x” on the plot.

Next, we analyzed DSSS modulation performance with fading channel effects. In wireless communication channel, there are multiple paths between transmitter and receiver and it has reflective effect on transferred signal. Such multipath fading effect result with receiving signal and its delated versions. Beside, signal is being exposed to scattering on a local scale in each path. Modeling such channel effect differs according to different multipath propagation scenarios. We used Rayleigh and Rician models for evaluating performance.

In Rayleigh fading channel model there are one or more major reflected paths from transmitter to receiver. BER vs SNR plot of DSSS is given below while signal is transmitted through Rayleigh fading channel.

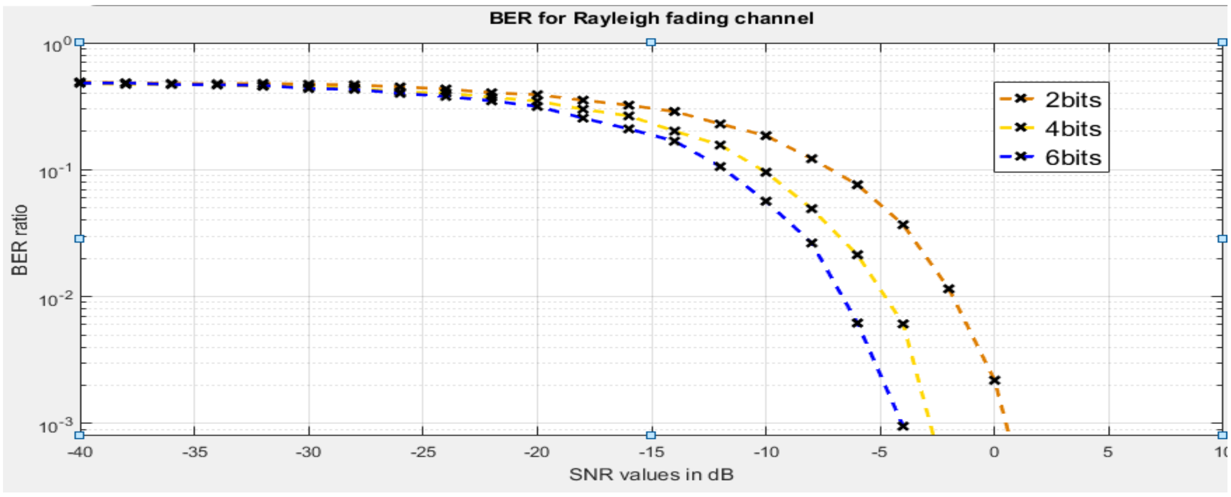

Figure 6: *Rayleigh fading channel performance*

Rician fading channel model is similar to Rayleigh, except it has direct line-of-sight path in addition. That is why, we expected better result compared to Rayleigh fading channel. BER vs SNR plot of DSSS is given below while signal is transmitted through Rician fading channel.

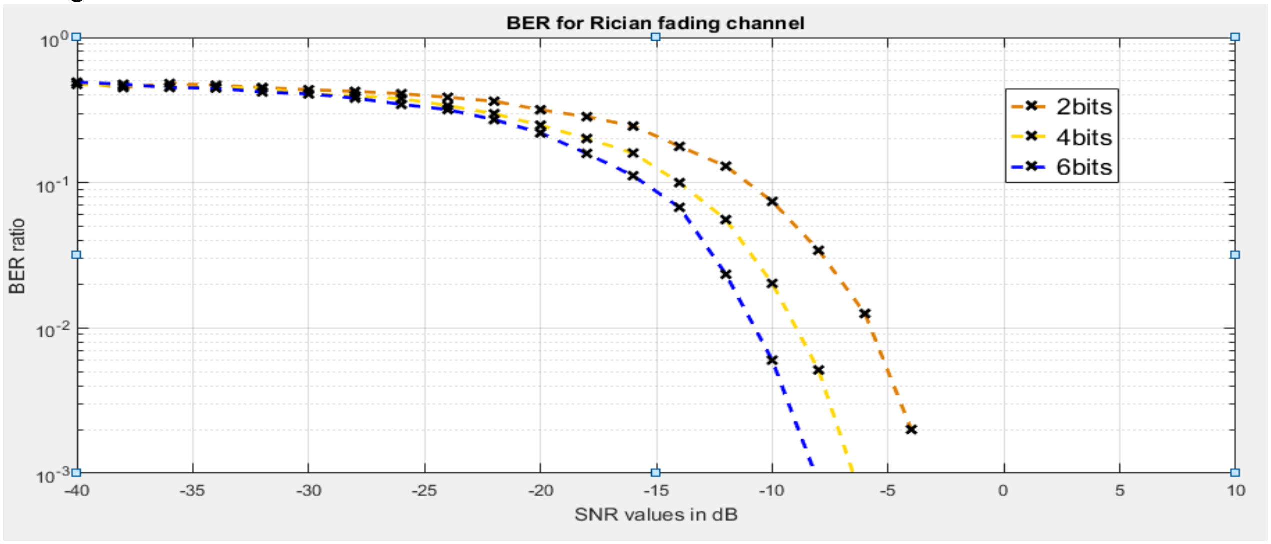

Figure 7: *Rician fading channel performance*

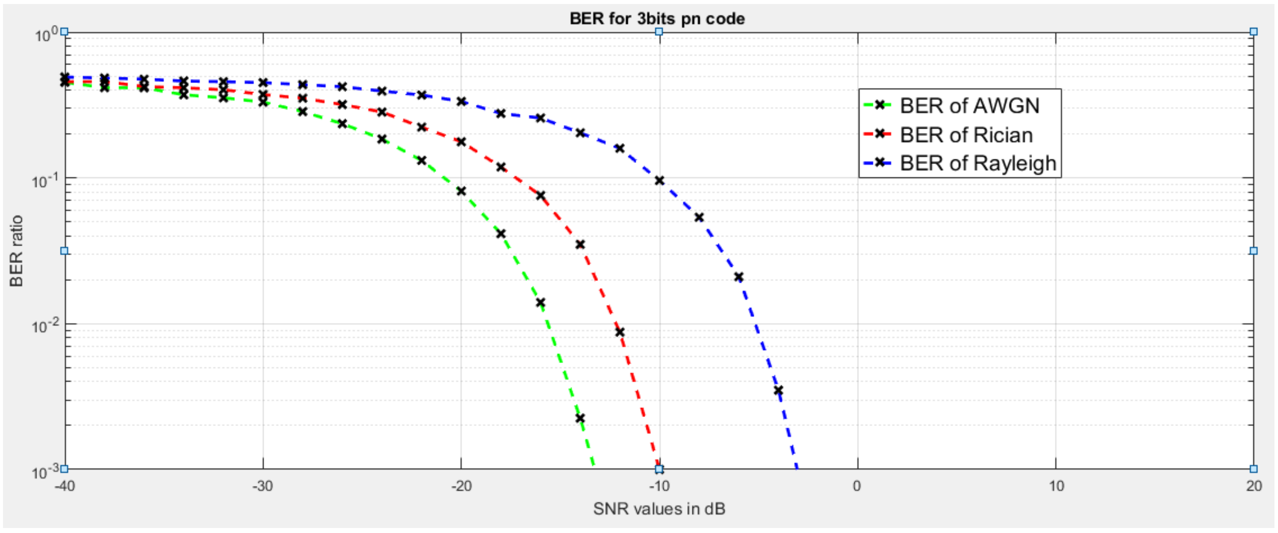

Figure 8: *Performance evaluation for all considered situations*

In the last plot, all of three conditions plotted in same graph while each message bit represented by 3 pseudo-random numbers.

D. Conclusion

In this project we work on end to end simulation of DSSS modulation. We used message signal modulated by BPSK modulation technique. Every message bit represented by different number of pseudo-number code for performance evaluation to see its effect clearly. We multiplied message signal and randomly generated pseudo-random sequence and obtained signal with spread spectrum. Because bandwidth increased, its resistance became strong against noise or channel fading effect. First we assumed that we transmitted and received signal without any channel effect. We demodulated received signal and obtained message signal. At the last step, we analyzed BER vs SNR plots by considering different situations which are Additive White Gaussian Noise, Rayleigh fading channel and Rician fading channel effects respectively. We repeated entire process 1000 times to draw Monte Carlo simulation distribution plots. While we increased number of pseudo-random code, we obtained better BER results. On the other hand, our modulation has higher resistance against Additive White Gaussian Noise and lower resistance against Rayleigh fading channel compared to others as expected.

References

1. https://people.cs.clemson.edu/~westall/851/spread-spectrum.pdf

2. A. Goldsmith, Wireless Communication. Stanford University Press, 2004. (p341)

Appendix 1

main.m

```
clear all;

%% parameters
Fs = 1000;
fc = 100;
fp = 2;
bit_t = 0.1;

%% message generation with BPSK
m = randi([0 1], 1, 20);
for bit = 1:length(m)
   if(m(bit)==0)
        m(bit) = -1;
   end
end

message =  repmat(m,fp,1);
message =  reshape(message,1,[]);

%% PN generation and multiply with message
pn_code = randi([0,1],1,length(m)*fp);

for bit = 1:length(pn_code)
   if(pn_code(bit)==0)
        pn_code(bit) = -1;
   end
end

DSSS = message.*pn_code;

%% create carrier and multipy with encoded sequence
t = 0:1/Fs:(bit_t-1/Fs);
s0 = -1*cos(2*pi*fc*t);
s1 = cos(2*pi*fc*t);
carrier = [];
BPSK = [];
for i = 1:length(DSSS)
    if (DSSS(i) == 1)
        BPSK = [BPSK s1];
    elseif (DSSS(i) == -1)
        BPSK = [BPSK s0];
    end
    carrier = [carrier s1];
end

Errors=zeros(1,10);

for h=1:10
    noise=[];
    p_error=0;
    for y=1:100
        noise=awgn(BPSK,-10*h);

%% demodulation
```

```
rx =[];
for i = 1:length(pn_code)
    if(pn_code(i)==1)
        rx = [rx noise((((i-1)*length(t))+1):i*length(t))];
    else
        rx = [rx  (-1)*noise((((i-1)*length(t))+1):i*length(t))];
    end
end

result = [];
for i = 1:length(m)
   x = length(t)*fp;
   cx = sum(carrier(((i-1)*x)+1:i*x).*rx(((i-1)*x)+1:i*x));
   if(cx>0)
       result = [result 1];
   else
       result = [result -1];
   end
end

counter=0;
for z=1:length(m)
    if m(z)~=result(z)
        counter=counter+1;
    end
end
p_error=p_error+counter*100/length(m);
    end
Errors(h) = (p_error/100);
end

%% Draw original message, PN code , encoded sequence on time domain
pn_size = length(pn_code);
tm  = 0:bit_t:(length(m)-1)*bit_t;
tpn = linspace(0,(pn_size-1)*bit_t/length(m),pn_size);

figure
subplot(311)
stairs(tm,m,'linewidth',2)
title('Message bit sequence')
axis([0 bit_t*length(m) -1 1]);
subplot(312)
stairs(tpn,pn_code,'linewidth',2)
title('Pseudo-random code');
axis([0 bit_t/length(m)*pn_size -1 1]);
subplot(313)
stairs(tpn,DSSS,'linewidth',2)
title('Modulated signal');
axis([0 bit_t/length(m)*pn_size -1 1]);


%% Draw original message, PN code , encoded sequence on frequency domain
f = linspace(-Fs/2,Fs/2,1024);
figure
subplot(321)
plot(f,abs(fftshift(fft(message,1024))),'linewidth',2);
title('Message spectrum')
subplot(322)
plot(f,abs(fftshift(fft(pn_code,1024))),'linewidth',2);
title('Pseudo-random code spectrum');
```

```
subplot(323)
plot(f,abs(fftshift(fft(DSSS,1024))),'linewidth',2);
title('Message multiplied by pseudo code');
subplot(324)
plot(f,abs(fftshift(fft(BPSK,1024))),'linewidth',2);
title('Transmitted signal spectrum');
subplot(325)
plot(f,abs(fftshift(fft(rx,1024))),'linewidth',2);
title('Received signal multiplied by pseudo code');
```

ber.m

```
clear all;

%% parameters
Fs = 1000;
fc = 100;
fp = 2;
bit_t = 0.1;
wq = 221;

nsnr = 25;
navg = 100;
snr_step = -2;
bit_error_1 = zeros(3,nsnr);
bit_error_2 = zeros(3,nsnr);
bit_error_3 = zeros(3,nsnr);

chan2 = ricianchan(1/Fs,0,3);
chan2.PathDelays = [0 1e-6 1e-7];
chan2.ResetBeforeFiltering = 0;
chan2.NormalizePathGains = 1;

for w = 1:3
%% message generation with BPSK
m = randi([0 1], 1, 20);
for bit = 1:length(m)
   if(m(bit)==0)
        m(bit) = -1;
   end
end

message =  repmat(m,fp,1);
message =  reshape(message,1,[]);

%% PN generation and multiply with message
pn_code = randi([0,1],1,length(m)*fp);

for bit = 1:length(pn_code)
```

```matlab
   if(pn_code(bit)==0)
        pn_code(bit) = -1;
   end
end

DSSS = message.*pn_code;

%% create carrier and multipy with encoded sequence
t = 0:1/Fs:(bit_t-1/Fs);
s0 = -1*cos(2*pi*fc*t);
s1 = cos(2*pi*fc*t);
carrier = [];
BPSK = [];
for i = 1:length(DSSS)
    if (DSSS(i) == 1)
        BPSK = [BPSK s1];
    elseif (DSSS(i) == -1)
        BPSK = [BPSK s0];
    end
    carrier = [carrier s1];
end

rice = filter(chan2,BPSK);

for h=1:nsnr
    noise   = [];
    BER_1 = 0;
    BER_2 = 0;
    BER_3 = 0;

    for y=1:navg

        noise = awgn(BPSK,(snr_step*h));
        rice_rx = awgn(rice,(snr_step*h));

         % Noise variance
        N0 = 1/10^((snr_step*h+11)/10);
        % Rayleigh channel fading
        ray_chan = 1/sqrt(2)*[randn(1,length(BPSK)) +
j*randn(1,length(BPSK))];
        % Send over Gaussian Link to the receiver
        rayl = ray_chan.*BPSK +
sqrt(N0/2)*(randn(1,length(BPSK))+i*randn(1,length(BPSK)));
        % Equalization to remove fading effects. Ideal Equalization
Considered
        rayl_rx = rayl./ray_chan;

        rx_1 =[];
        rx_2 =[];
        rx_3 =[];

        for i = 1:length(pn_code)
            if(pn_code(i)==1)
                rx_1 = [rx_1 noise((((i-1)*length(t))+1):i*length(t))];
                rx_2 = [rx_2 rayl_rx((((i-1)*length(t))+1):i*length(t))];
                rx_3 = [rx_3 rice_rx((((i-1)*length(t))+1):i*length(t))];
            else
                rx_1 = [rx_1 (-1)*noise((((i-
1)*length(t))+1):i*length(t))];
```

```
                rx_2 = [rx_2 (-1)*rayl_rx((((i-
1)*length(t))+1):i*length(t))];
                rx_3 = [rx_3 (-1)*rice_rx((((i-
1)*length(t))+1):i*length(t))];
            end
        end

        result_1 = zeros(1,length(m));
        result_2 = zeros(1,length(m));
        result_3 = zeros(1,length(m));
        for i = 1:length(m)
           x = length(t)*fp;
           cx_1 = sum(carrier(((i-1)*x)+1:i*x).*rx_1(((i-1)*x)+1:i*x));
           cx_2 = sum(carrier(((i-1)*x)+1:i*x).*rx_2(((i-1)*x)+1:i*x));
           cx_3 = sum(carrier(((i-1)*x)+1:i*x).*rx_3(((i-1)*x)+1:i*x));
           if(cx_1>0)
               result_1(i) =  1;
           else
               result_1(i) = -1;
           end

           if(cx_2>0)
               result_2(i) =  1;
           else
               result_2(i) = -1;
           end

           if(cx_3>0)
               result_3(i) =  1;
           else
               result_3(i) = -1;
           end
        end

        counter_1 = 0;
        counter_2 = 0;
        counter_3 = 0;
        for z=1:length(m)
            if m(z)~=result_1(z)
                counter_1 = counter_1+1;
            end

            if m(z)~=result_2(z)
                counter_2 = counter_2+1;
            end

            if m(z)~=result_3(z)
                counter_3 = counter_3+1;
            end
        end
        BER_1 = BER_1 + counter_1/length(m);
        BER_2 = BER_2 + counter_2/length(m);
        BER_3 = BER_3 + counter_3/length(m);
    end
    bit_error_1(w,h) = (BER_1/navg);
    bit_error_2(w,h) = (BER_2/navg);
    bit_error_3(w,h) = (BER_3/navg);
end

fp = fp+2;
```

```
end

%% draw BER - SNR graph
time_vec = (10):snr_step:((nsnr-1)*snr_step+10);
figure;
subplot(221)
semilogy(time_vec,bit_error_1(1,:),'g--o',time_vec,bit_error_1(2,:),'r--
o',time_vec,bit_error_1(3,:),'b--o')
legend('2bits', '4bits','6bits')
axis([ -40 10 0 1 ])
title(['BER for awgn noise']);
xlabel('SNR values in dB');
ylabel('BER ratio');
subplot(222)
semilogy(time_vec,bit_error_2(1,:),'g--o',time_vec,bit_error_2(2,:),'r--
o',time_vec,bit_error_2(3,:),'b--o')
legend('2bits', '4bits','6bits')
axis([ -30 30 0 1 ])
title(['BER for Rayleigh channel']);
xlabel('SNR values in dB');
ylabel('BER ratio');
subplot(223)
semilogy(time_vec,bit_error_3(1,:),'g--o',time_vec,bit_error_3(2,:),'r--
o',time_vec,bit_error_3(3,:),'b--o')
legend('2bits', '4bits','6bits')
axis([ -40 10 0 1 ])
title(['BER for Rician channel']);
xlabel('SNR values in dB');
ylabel('BER ratio');
subplot(224)
semilogy(time_vec,bit_error_1(3,:),'g--o',time_vec,bit_error_2(3,:),'r--
o',time_vec,bit_error_3(3,:),'b--o')
legend('BER of AWGN', 'BER of Rayleigh','BER for Rician')
axis([ -30 30 0 1 ])
title(['BER for 6bits pn code ']);
xlabel('SNR values in dB');
ylabel('BER ratio');
```