We simulate uncoded BER of BPSK modulated data as a function of SNR
- in an AWGN channel
- in a Rayleigh fading channel
- in an AWGN channel when direct sequence spreading is used
and compare results to the theoretical ones.
We assume coherent receiver and perfect synchronization.

%set used SNR values
%SNR (Eb/No) values in decibels
SNR=[0:2:14]'; %column vector
%SNR in linear scale
snr=10.^(SNR/10);

%we create initial zero vectors for BER
BER1=zeros(length(SNR),1);
BER2=BER1;
BER3=BER1;
% we need a DS-code, we create a random, complex one, length Nc
% elements +-1 +- j*1
Nc=32;
% note that all parameters are defined as variables
% their change afterwards is easy
%(no need to change it every place, just once)
ds=(2*round(rand(Nc,1))-1)+j*(2*round(rand(Nc,1))-1); % ds-code

% plot the ds signal
plot([real(ds) imag(ds)]), axis([0 Nc -1.1 1.1])
title('real and imaginary parts of DS code'), legend('real','imag'), pause
-----------------------------------------------

% we use symbol energy normalized to 1
% thus, DS energy is normalized to 1 (it is a pulse waveform)
ds=ds/norm(ds);
% check this
ds_energy=norm(ds)^2, pause
(NOTE: normalization is a usual trick)
-----------------------------------------------

% Monte Carlo loop starts here
% some initial values
% totally Nmax symbols
Nmax=1000; % maximum number of iterations
Nerr=100; % minimum number of errors

for k=1:length(SNR), % we do MC trials for each SNR
 for l=1:Nmax, % MC loop
%DATA
% we create data as vectors of length Ns symbols
% and thus use MATLAB's vector processing capabilities
% in addition to for loops (since too long vectors are problems
% to some versions of MATLAB)
Ns=100;
data=2*round(rand(Ns,1))-1;
data is random and generated again and again for each MC trial
% totally Ns * Nmax symbols, if 100*1000 = 100 000
% results rather reliable down to 1e-4
% plot data
if l==1 & k==1, % we plot/see things only once, at the first round
plot(data),title('data'),axis([0 Ns -1.1 1.1]),pause,
end
%MODULATION
%BPSK signal
bpsk=data;

%DS spread signal
DS=kron(data,ds); %length Ns*Nc
( kron: Kronecker product, element matrices of
kron(A,B) are A(i,j)B )

%plot first 2 symbols of ds-modulated signal
if l==1 & k==1
plot([real(DS) imag(DS)]),title('2 symbols of DS
modulated signal'),
axis([0 2*Nc -1/sqrt(Nc) 1/sqrt(Nc)]),pause,
end

%CHANNELS
%This is the place to set SNR.
%Since symbol energy is 1 and noise variance is 1,
%SNR of symbol/noise sample is 0 dB.
%Thus, we have to multiply symbol or divide noise
to obtain desired SNR.
%Since snr is power variable we have to
multiply/divide by
%sqrt(snr) to have amplitude coefficient.
% noise generation
% for BPSK
n = 1/sqrt(2) * (randn(Ns,1) + j*randn(Ns,1));
% Since complex noise is generated by two real noise
sequences the
% total variance is 2 x variance of one sequence
(namely 1). If we multiply
% by 1/sqrt(2) the total variance of noise becomes
1.
% This is two sided noise spectral density No in BER
equations.

% we check this
if l==1 & k==1,
    var_n=norm(n)^2, pause
end
% This should be Ns since we sum Ns variables.
% Since n is a realization of a random process,
% the result is not exact.
% Average of repeated trials gives more exact
result.

% noise for DS-BPSK
% Since signal is longer (by factor Nc) we need
different noise.
n2 = 1/sqrt(2) * (randn(Ns*Nc,1) + j*randn(Ns*Nc,1));
% noise is random and we generate it again and again
for each MC trial
%AWGN BPSK

Bpsk=sqrt(snr(k))*data+n;  %snr is Eb/N0 in BER equations
if l==1 & k==1,
  plot([real(Bpsk) data])
  legend('real part of signal','data'),
  title('BPSK signal in noise'),pause
end

---------------------------------------------

%AWGN DS-BPSK

Ds=sqrt(snr(k))*DS+n2;
(this works since DS energy is normalized to 1)

---------------------------------------------

%Rayleigh fading BPSK signal

%first we create taps for each symbol
  taps=1/sqrt(2)*(randn(Ns,1)+j*randn(Ns,1));
%these are zero mean unit variance complex Gaussian variables
  Bpsk_r=sqrt(snr(k))*abs(taps).*data+n;  %SIGNAL
%notice usage of elementwise vector or matrix product .*

if l==1 & k==1,
  plot([real(Bpsk_r) data])
  legend('real part of signal','data'),
  title('BPSK signal in noise & fading channel'),pause
end

---------------------------------------------
%difference between AWGN and Rayleigh channel
if l==1 & k==1,
plot(abs([Bpsk Bpsk_r]))
legend('AWGN','RAYLEIGH'),
title('BPSK in AWGN & Rayleigh fading channel'),pause
end
%variations on envelope are larger in fading channels
%deeper nulls and also higher peaks

%DEMODULATION
%you have to know how these signals are demodulated
%coherent + synchronized reception

%BPSK
r1=real(Bpsk); %demodulated signal, soft decision
%because phase is 0, if phase is h
%r1=real(Bpsk*exp(-j*2*pi*h)); i.e., phase is cancelled

%BPSK in fading channel
r2=real(Bpsk_r);
% DS-BPSK
here we need MF to the DS code before taking the real part
different ways to do it
we select correlation approach where each code length block is
multiplied by complex conjugate of the code
for v=1:Ns, %we have Ns blocks
r3(v)=real(ds'*Ds((v-1)*Nc+1:v*Nc));
end
r3=r3(:);
---------------------------------------------------

demodulated symbols are actually MF output peaks
% separated by Nc samples
if l==1 & k==1,
plot(real(filter(conj(ds(Nc:-1:1)),1,DS)))
legend('without noise')
title('DS-BPSK signal after MF over 6 symbols'),
axis([0 6*Nc -3 3]),pause
end
---------------------------------------------------
%different demodulated symbols
if l==1 & k==1,
plot([r1 r2 r3])
legend('AWGN','Rayleigh','DS'),
title('demodulated symbols'),pause
end
%BPSK and DS are close (as they should be)
%we can run this at high SNR to see it closely
---------------------------------------------

%hard decisions, converts soft demodulated symbols to sequence of +-1
%AWGN
d1=find(r1>=0);d2=find(r1<0);
r1(d1)=1;r1(d2)=-1;
%Rayl
d1=find(r2>=0);d2=find(r2<0);
r2(d1)=1;r2(d2)=-1;
%DS
d1=find(r3>=0);d2=find(r3<0);
r3(d1)=1;r3(d2)=-1;

%plot example
if l==1 & k==1,
plot([r1 r2 r3])
legend('AWGN','Rayleigh','DS'),
axis([0 Ns -1.1 1.1]),
title('demodulated symbols after hard decisions')
pause
end
---------------------------------------------
%BER analysis
%errors in the current MC run
Ber1=length(find((data-r1)~=0)); %number of errors in AWGN
Ber2=length(find((data-r2)~=0)); %number of errors in Rayleigh
Ber3=length(find((data-r3)~=0)); %number of errors in DS-BPSK
if k==1 & l==1,
    errors=[Ber1 Ber2 Ber3],pause
end

%we add errors to previous error counts, initially zero
%index k is for SNRs
BER1(k)=BER1(k)+Ber1; %AWGN
BER2(k)=BER2(k)+Ber2; %Rayleigh
BER3(k)=BER3(k)+Ber3; %DS-BPSK

%we stop MC trials if minimum number of errors is obtained in all systems
if BER1(k)>Nerr & BER2(k)>Nerr & BER3(k)>Nerr,
    break %terminates the innermost loop
end

end %end MC
we calculate BER by dividing number of successful trials
by their total number
BER1(k)=BER1(k)/Ns/l;
BER2(k)=BER2(k)/Ns/l;
BER3(k)=BER3(k)/Ns/l;

end %ends SNR loop

all simulated BERs and corresponding SNR in a matrix
BER=[SNR BER1 BER2 BER3]

finally we compute theoretical values and compare them to simulation results

AWGN BER is function of sqrt(2*SNR)
The_awgn=.5*erfc(2*sqrt(snr)/sqrt(2));
Rayleigh BER is different function of SNR
The_rayl=.5*(1-sqrt(1+snr)/snr));
%note elementwise division ./
%logarithmic plot (y-axis)
semilogy(SNR,[The_awgn The_rayl BER1 BER2 BER3])
xlabel('SNR [dB]')
ylabel('BER')
axis([0 SNR(length(SNR)) 1e-6 .5])
grid on
legend('Theor AWGN','Theor Rayl.','AWGN','Rayl.','DS-BPSK')

%simulated and theoretical results are close,
especially if total number of
%symbols is large enough

%BPSK and DS-BPSK perform similarly (as they should now)

%Rayleigh fading channel is much more difficult
environment than AWGN
%over 10 dB extra power is needed in transmission
to have equal results (BER)
%10 dB more = 10 times higher power, e.g., from 1 W
to 10 W