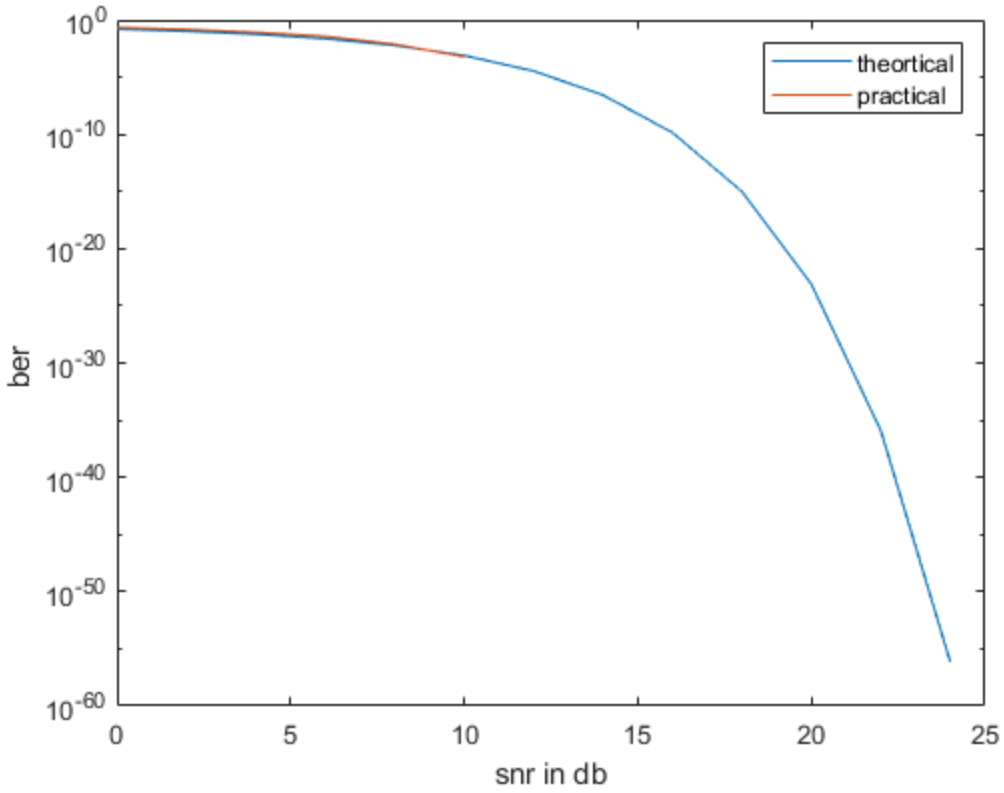
```
clc;
clear all;
close all;

N= 10000;

% r =[0 0 0 1 1 0 1 1];
% N = length(r);
r = randi([0,1],1,N);
k =1;
for i=1:2:N                     %converted to qpsk

        if((r(i)==0)&&(r(i+1)==0))
            r1(k) = 1+1j;

        elseif((r(i)==0)&&(r(i+1)==1))
            r1(k) = -1+1j;

        elseif((r(i)==1)&&(r(i+1)==0))
            r1(k) = -1-1j;

        elseif((r(i)==1)&&(r(i+1)==1))
            r1(k) = 1-1j;

        end
    k = k+1;

end

n = randn(1,N/2)+j*randn(1,N/2);


snr_db = 0:2:24;
kk=1;% snr in db

for k = 1: length(snr_db)

snr_linear = 10.^(snr_db(k)/10);          % converted snr to linear
sigma = 1./(snr_linear).^(1/2);               % find sigma

y = r1+ sigma.*n;                % find y = bpsk_signal + sigma*noise


% convert y sequence into bpsk take threshold value = 0
% z is your constructed signal
% y is output signal with noise
for j=1:N/2
    a=real(y(j));
    b = imag(y(j));
   if((a>=0)&&(b>=0))
            z(2*j-1)=0;
            z(2*j)=0;
```

```
   elseif((a<0)&&(b>=0))
            z(2*j-1)=0;
            z(2*j)=1;

   elseif((a<0)&&(b<0))
            z(2*j-1)=1;
            z(2*j)=0;

   elseif((a>=0)&&(b<0))
            z(2*j-1)=1;
            z(2*j)=1;
   end
end

% check bit by bit that r and z is same or not

count = sum(r ~= z);

% calculate theortical ber  Q((snr_linear)^(1/2)


ber_th(k) = qfunc((snr_linear).^(1/2));
ber_prac(k) = count/N;
%k=k+1;
end
%snr_linear1 = 10.^(snr_db/10)  ;

semilogy(snr_db, ber_th);
% xlabel("snr in db");
% ylabel("ber theortical");

hold on
semilogy(snr_db, ber_prac);
xlabel("snr in db");
ylabel("ber ");


legend('theortical','practical')
```



Published with MATLAB® R2018b