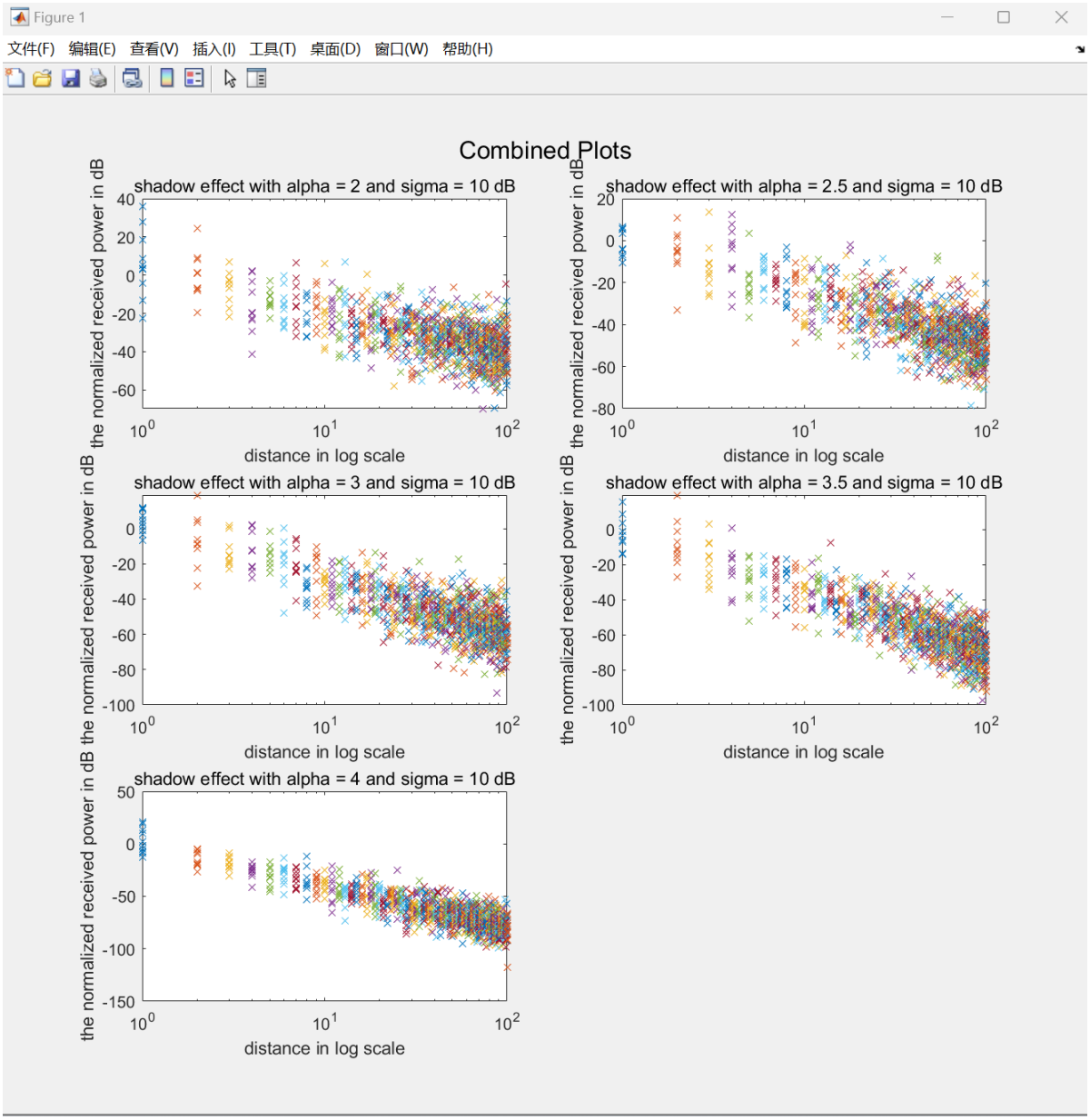
1,

a) simulate pr/p0 in dB scale with noise

```
命令行窗口
  ------------------------------------------------------------------------
  -------------------------class practice 2.1----------------------------
  -------------------by Amber-GaoQi on 23/7/26---------------------
  ------------------------------------------------------------------------
  distance:1:1:100
  standard deviation of shadowing in dB:10
  the number of samples for each distance:10
  the pathloss exponent,alpha:2:0.5:4
fx >>
```

b) the results



We can see that despite the noise interference, the overall trend still fits the formula, and the slope decreases with increasing alpha.

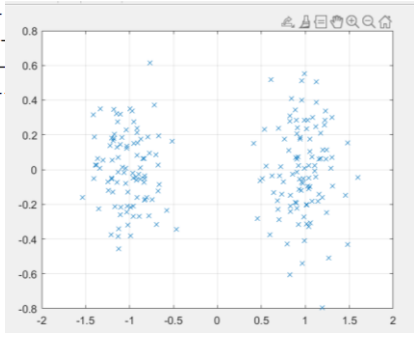
c) the program

```
function classpractice21()

clc
clear all

disp('------------------------------------------------------------------------')
disp('-------------------------class practice 2.1---------------------------')
disp('--------------------by Amber-GaoQi on 23/7/26---------------------')
disp('------------------------------------------------------------------------')

% Input parameters from the user
D = input('distance:');   % 1:1:100
sigma = input('standard deviation of shadowing in dB:');
N = input('the number of samples for each distance:');
alpha = input('the pathloss exponent,alpha:'); % 2:0.5:4

figure; % Create a single figure outside the loops

% Create subplots arrangement based on the number of alpha values
num_alphas = length(alpha);
num_rows = ceil(sqrt(num_alphas));
num_cols = ceil(num_alphas / num_rows);

for a = 1:num_alphas
    for m = 1:length(D)
        for n = 1:N
            % Calculate The ratio of PR to P0 in dB scale with shadowing for each distance
            PrDB(m, n) = -10 * alpha(a) * log10(D(m)) + randn * sigma;
                                        %RANDN produces a standard normal distribution
        end
    end

    % Create a subplot for each alpha value
    subplot(num_rows, num_cols, a);

    % Plot each set of data on the subplot
    for m = 1:length(D)
        semilogx(D(m) * ones(1, N), PrDB(m, :), 'x')
        hold on
    end

    xlabel('distance in log scale')
    ylabel('the normalized received power in dB')
    str = sprintf('shadow effect with alpha = %g and sigma = %g dB', alpha(a), sigma);
    title(str);
end

% Adjust the layout to make the subplots fit nicely
sgtitle('Combined Plots'); % Add a common title for the combined figure

end
```

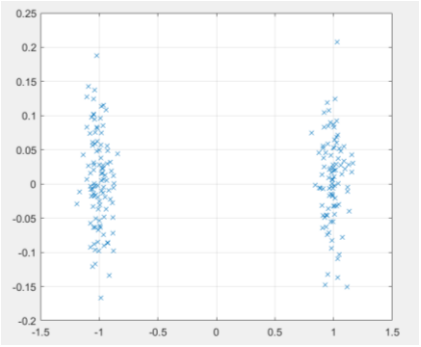
2,

a)

```
--------------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27---------------------
--------------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 0
the number of random samples: 200
average SNR in dB: 10
>>
```



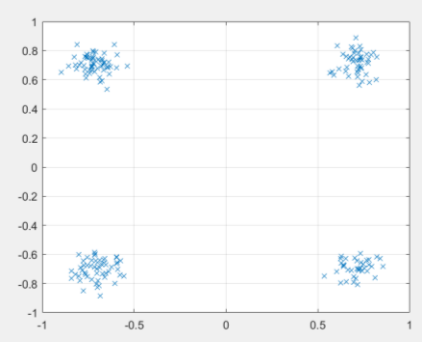
```
--------------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27---------------------
--------------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 0
the number of random samples: 200
average SNR in dB: 20
```



The smaller the SNR, the greater the impact of noise

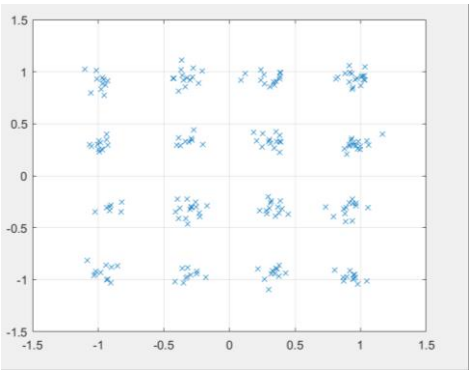
b)

```
--------------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27---------------------
--------------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 1
the number of random samples: 200
average SNR in dB: 20
```



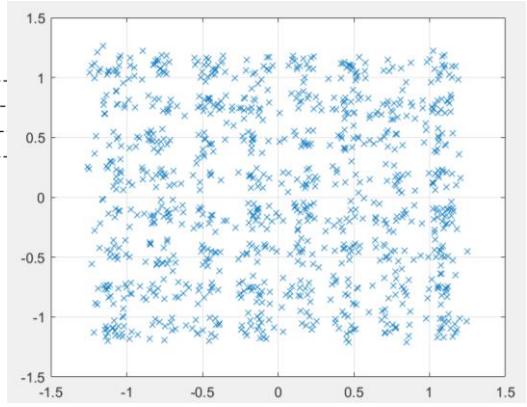
c)

```
--------------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27---------------------
--------------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 2
the number of random samples: 200
average SNR in dB: 20
```

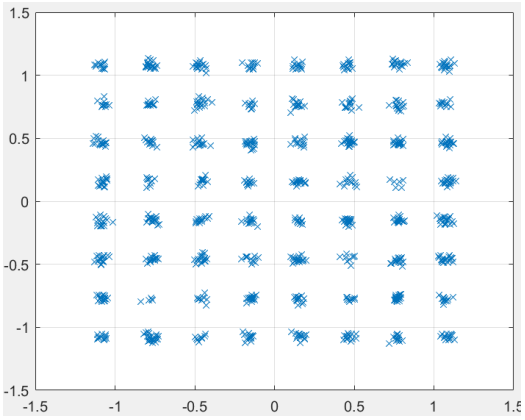


d)

```
------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27--------------------------
------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 3
the number of random samples: 1000
average SNR in dB: 20
>>
```




```
------------------------------------------------------------------------
------------------------class practice 2.2------------------------------
-------------------by Amber-GaoQi on 23/7/27--------------------------
------------------------------------------------------------------------
0-BPSK
1-QPSK
2-16-QAM
3-64-QAM

choose 0, 1, 2 or 3: 3
the number of random samples: 1000
average SNR in dB: 30
```



e) the program

```
function classpractice22()
clc
clear all

disp('------------------------------------------------------------------')
disp('-------------------------class practice 2.2-------------------------')
disp('-------------------by Amber-GaoQi on 23/7/27--------------------')
disp('------------------------------------------------------------------')


disp('0-BPSK')            %1 bit per symbol
disp('1-QPSK')            % 2 bits per symbol
disp('2-16-QAM')       % 4 bits per symbol
disp('3-64-QAM')       % 6 bits per symbol

fprintf('\n')

Mod=input('choose 0, 1, 2 or 3: ');
N=input('the number of random samples: '); %200,1000
snrdB=input('average SNR in dB: '); %10,15,20,30
snr=10^(snrdB/10);
p=1;
sigma=sqrt(p/snr); %snr=p/sigma^2

switch Mod
    case 0%BPSK
        c=[-1 1];
        s=c(randi(2,1,N));
        noise = (randn(1,N)+1i*randn(1,N))/sqrt(2)*sigma;
        y=s+noise;
        plot(y,'x')
        grid on
    case 1%QPSK
        c=[-1-1i -1+1i 1-1i 1+1i];
        pav=mean(abs(c).^2);
        c=c/sqrt(pav);
        s=c(randi(4,1,N));
        noise=(randn(1,N)+1i*randn(1,N))/sqrt(2)*sigma;
        y=s+noise;
        plot(y,'x')
        grid on
    case 2%16-QAM
        sI=[-3 -1 1 3 ];
        sQ=[-3 -1 1 3];
        k=1;
        for m=1:4
            for n=1:4
                c(k)=sI(m)+1i*sQ(n);
                k=k+1;
            end
        end
        pav=mean(abs(c).^2);
        c=c/sqrt(pav);
        s=c(randi(16,1,N));
        noise=(randn(1,N)+1i*randn(1,N))/sqrt(2)*sigma;
        y=s+noise;
        plot(y,'x')
        grid on
    case 3%64-QAM
        sI=[-7 -5 -3 -1 1 3 5 7];
        sQ=[-7 -5 -3 -1 1 3 5 7];
        k=1;
        for m=1:8
            for n=1:8
```

```
            c(k)=sI(m)+1i*sQ(n);
            k=k+1;
        end
    end
    pav=mean(abs(c).^2);
    c=c/sqrt(pav);
    noise=(randn(1,N)+1i*randn(1,N))/sqrt(2)*sigma;
    s=c(randi(64,1,N))+noise;
    plot(s,'x')
    grid on
end
end
```

3,

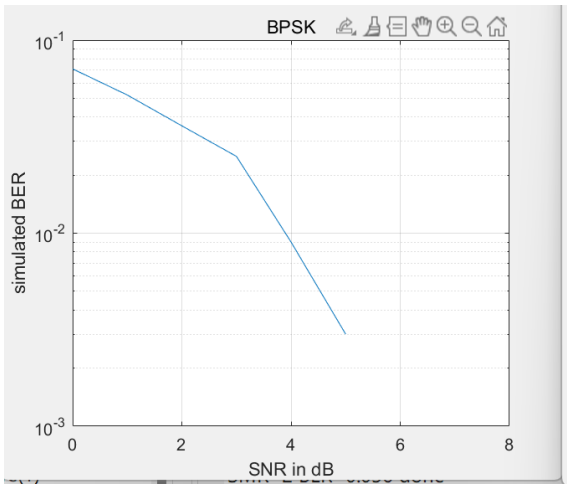
a)

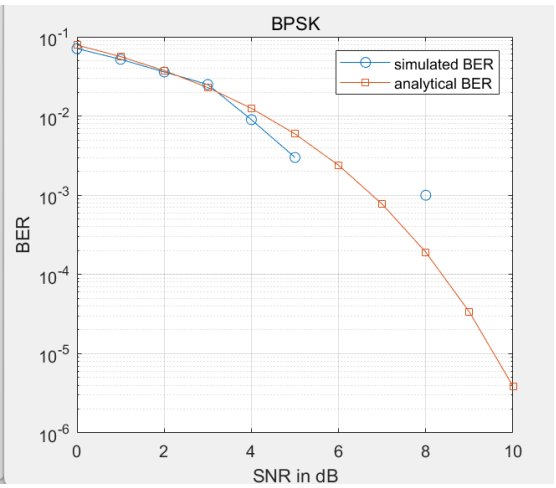
```
function classpractice23()
clc
clear

disp('------------------------------------------------------------------------')
disp('-------------------------class practice 2.3-----------------------------')
disp('--------------------by Amber-GaoQi on 23/7/26-----------------------')
disp('------------------------------------------------------------------------')

snrdB=input('the range of SNR in dB for simulations: ');%0:1:10
snr=10.^(snrdB/10);
A=1;
sigma = sqrt(A^2/2./snr);

Nsim=input('the number of random samples: ')

tic

c=[-1 1];   %A=1, BPSK

for m=1:length(snr)
    Nerror(m)=0;    %initial the error

    for n=1:Nsim
        s(n)=c(randi(2));
        noise(n)=randn*sigma(m);
        y(n)=s(n)+noise(n);

        s_det(n)=(y(n)>0)*1+(y(n)<=0)*(-1); %get a new s after receive s with noise
        %use matrices to avoid for loop
        %if y(n)>0
        %   s_det(n)=1;
        %else
        %   s_det(n)=-1;
        %end

        if s(n)~=s_det(n) %there's a mistake
            Nerror(m)=Nerror(m)+1;
        end
    end

    BER(m)=Nerror(m)/Nsim; %average rate
    fprintf('SMR=%g BER=%g done\n', snrdB(m),BER(m))
end

toc

figure(1)
semilogy(snrdB,BER); %draw the plot
xlabel('SNR in dB')
ylabel('simulated BER')
title('BPSK')
grid on;

BER_anal=analytical_ber(snrdB);

%campare two plots
figure(2);
semilogy(snrdB,BER,'o-');
hold on
semilogy(snrdB,BER_anal,'s-');
xlabel('SNR in dB')
ylabel('BER')
legend('simulated BER','analytical BER')
title('BPSK')
grid on;

end
```

```
function BER=analytical_ber(snrdB)

snr=10.^(snrdB/10);
L=length(snr);

for m=1:L
    BER(m)=erfc(sqrt(snr(m)))/2;
end

end
```

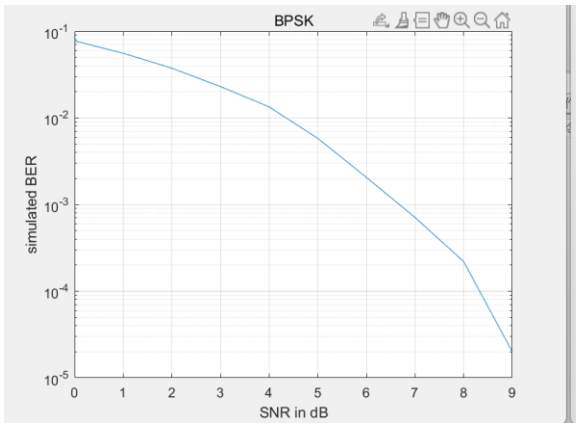
b)

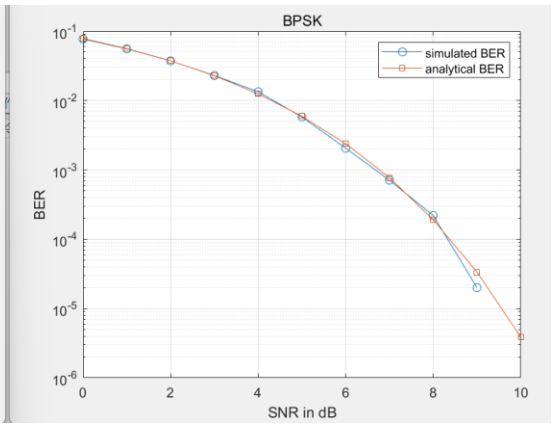
```
--------------------------------------------------------------------
------------------------class practice 2.3----------------------------
-------------------by Amber-GaoQi on 23/7/26---------------------
--------------------------------------------------------------------
the range of SNR in dB for simulations: 0:10
the number of random samples: 1000
```





then enlarge the number of samples:

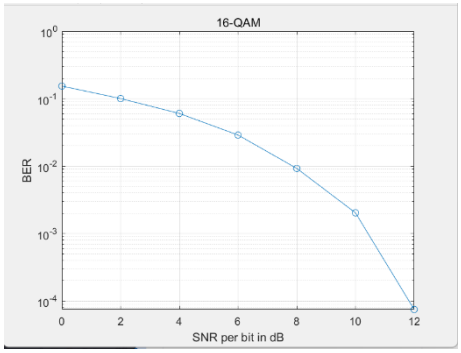




4,

a) the result

```
--------------------------------------------------------------------
------------------------class practice 2.4----------------------------
-------------------by Amber-GaoQi on 23/7/26---------------------
--------------------------------------------------------------------
choose the mod 0,1,2 or 3: 2
the range of SNR in dB for simulations: 0:2:20
the number of random samples: 10000
```



b) the program

```
function classpractice24()
clc
clear

disp('------------------------------------------------------------')
disp('-------------------------class practice 2.4-------------------')
disp('-------------------by Amber-GaoQi on 23/7/26-----------')
disp('------------------------------------------------------------')

mTYPE=input('choose the mod 0,1,2 or 3: ');
snrdB_bit=input('the range of SNR in dB for simulations: ');
snr_bit=10.^(snrdB_bit/10);
Nsim=input('the number of random samples: ');


switch mTYPE
    case 0
        NofP=2; %the number of constellation points
    case 1
        NofP=4;
    case 2
        NofP=16;
    case 3
        NofP=64;
end
snr=snr_bit*log2(NofP);
L=length(snr_bit);
P=1;
sigma=sqrt(P./snr);

for m=1:L
    [s,c]=mod_symbols(Nsim,mTYPE);
    noise=(randn(1,Nsim)+1i*randn(1,Nsim))/sqrt(2)*sigma(m);
    y=s+noise;
```

```
    if mTYPE==0
        y=real(y);  %BPSK
    end

    Nerror(m)=0;

    %get the simbol error rate
    for n=1:Nsim...
    ser(m)=Nerror(m)/Nsim
    ber(m)=1-(1-ser(m))^(1/log2(NofP));
    ber1(m)=ser(m)/log2(NofP); %another way of calculate

    fprintf('SNR=%gdB and BER=%g done \n',snr_bit(m),ber(m));
end

semilogy(snrdB_bit,ber,'-o');
grid on
xlabel('SNR per bit in dB')
ylabel('BER')
switch mTYPE
    case 0
        title('BPSK')
    case 1
        title('QPSK')
    case 2
        title('16-QAM')
    case 3
        title('64-QAM')
end
end
```

c）the functon mod_symbols

```
function [s,c]=mod_symbols(Nsim,mTYPE)

switch mTYPE
    case 0
        c=[-1 1];
        NofP=2;
    case 1
        c=[-1-1i -1+1i 1-1i 1+1i];
        NofP=4;
    case 2
        sI=[-3 -1 1 3 ];
        sQ=[-3 -1 1 3];
        NofP=16;
    case 3
        sI=[-7 -5 -3 -1 1 3 5 7];
        sQ=[-7 -5 -3 -1 1 3 5 7];
        NofP=64;
end

if mTYPE~=0
    k=1;
    for m=1:length(sI)
        for n=1:length(sQ)
            c(k)=sI(m)+1i*sQ(n);
            k=k+1;
        end
    end
    Pav=mean(abs(c).^2);
    c=c/sqrt(Pav);
end
s=c(randi(NofP,1,Nsim));
end
```