Homework #1, AA2021-2022: MIMO System Identification and Deconvolution

1.1 Noise generation

-Sect.1 Theoretical tools and methods adopted to solve the problem

For this problem I evaluated the sample covariance using “Cholevskiy method” and applied Monte Carlo simulation discussed during MATLAB exercises. In order to evaluate the closure of the matrices, sum of squared errors(SSE) was applied.

-Sect.2 Solution

```
%defined metric SSE (SUM OF SQUARE ERRORS)
K = floor(linspace(10, 200,10));%numbere of samples
rho = linspace(0, 0.99, 10);
M = 4;
Nmc = 2*10^4;
% Noise parameters:
C = zeros(M,M);
SSE = zeros(length(K), length(rho), Nmc);% Metric adopted - SUM OF SQUARED ERRORS

for index_K = 1:length(K)
for index_rho =1:length(rho)%different rho values
for n =1:Nmc % Monte-Carlo simulation
C = Cov_True(rho(index_rho), M);%TRUE COV
U = randn(M, K(index_K)); % White noise
w_s = chol(C,"lower")* U;% Cholesky method
C_s = cov(w_s.');%SAMPLE COV
%%SSE%%%
SSE(index_K,index_rho) =  SSE(index_K,index_rho) +
sum(sum((C_s-C).^2))/Nmc;
end
end
end

figure;
grid on; box on;
semilogy(K, mean(SSE,3))
title('1.1 SSE vs K')
xlabel('K')
ylabel('SSE')
lgd = legend(string(rho));
title(lgd,'\rho')
```

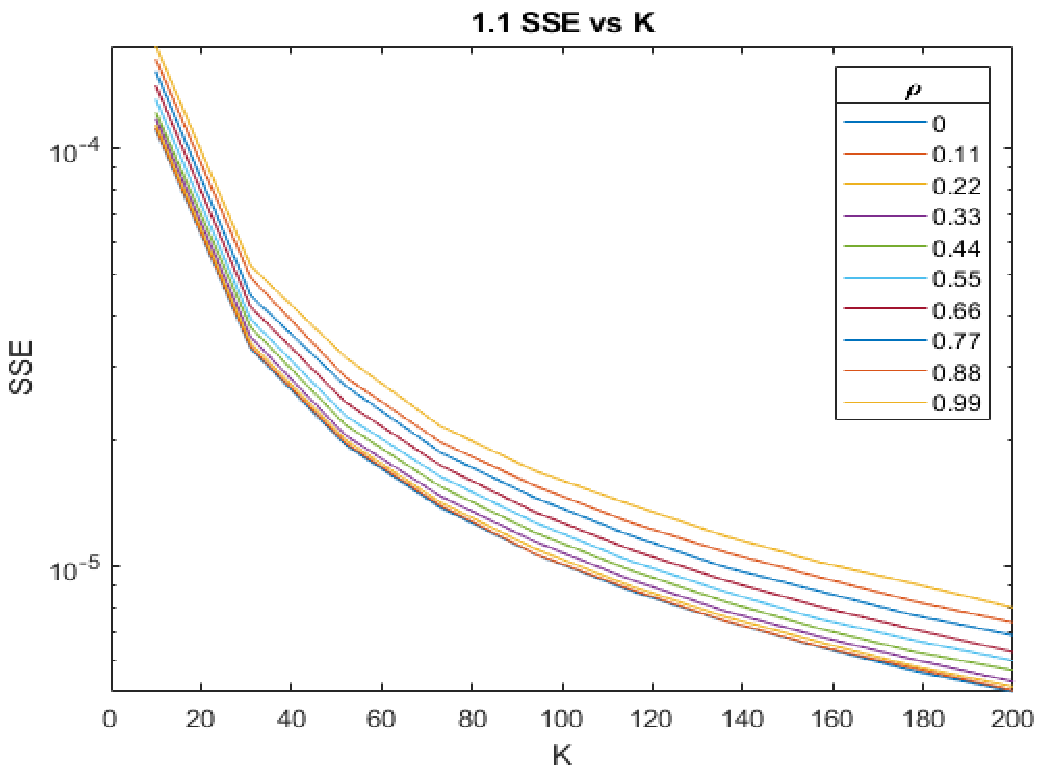


Fig. 1.1 Dependency of SSE on the number of samples for different values of ρ

-Sect.3 Conclusion

As expected, the larger number of samples we have, the beter the estimate is. In addition we can say that the lower value of ρ correspond to the SSE for the same number of samples K. For instance, for K = 94, in case ρ = 0 we the SSE is approximately 0.21, whereas for ρ = 0.99, we get 0.35.

1.2 MIMO estimation

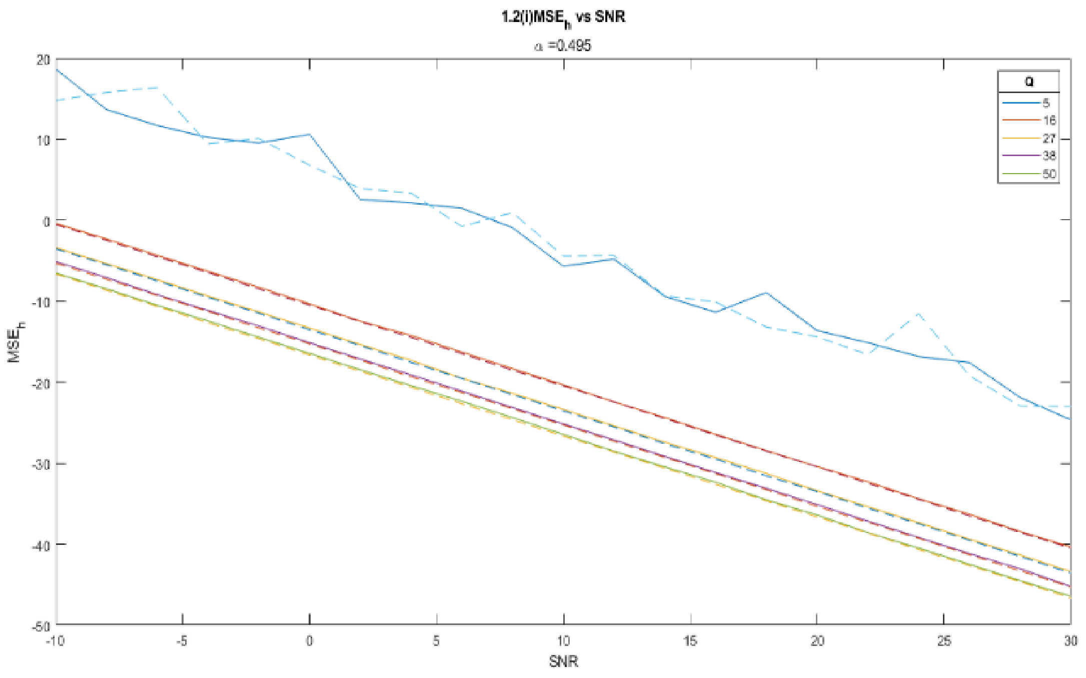
-Sect.1 Theoretical tools and methods adopted to solve the problem

For this problem MLE was used in order to estimate the system. After that to estimate the performance, represented graphically MSE. Having done that, CRB was plotted to evaluate the results. Additionally, Kronoker product and Monte Carlo simulation were applied.

-Sect.2 Solution

```
%% 1.2(i) MIMO estimation
%clear all; clc;
disp('===1.2(i)===')
M = 4;
N = 4;
SNR_dB = -10:2:30; %dB
SNR = 10.^(SNR_dB/10);%SNR_linear
alpha = linspace(0,0.99,5);
Q = floor(linspace(5, 50, 5));%Q_min>=K(M-1)+1
rho = 0.1;
Nmc = 1000;%2*10^3;
MSE_h= zeros(length(Q), length(alpha) , length(SNR), Nmc);
CRB = zeros (length(Q), length(SNR), Nmc);

for Q_index = 1:length(Q)
    disp( ['Q =', num2str(Q(Q_index)), '/50'])
    for a = 1:length(alpha)%different alpha
        h = zeros(M, N);
        for l= 1:N
            for i = 1:M
                h(i,l) = alpha(a)^abs(i-l);
            end
        end

        for s = 1:length(SNR)
            for p =1 :Nmc%MC
                x = randn(Q(Q_index),N);
                c = Cov_True(rho, M)/SNR(s);
                w = chol(c, 'lower')* randn(M, Q(Q_index));

                W = (reshape(w',1,[]))';
                H = (reshape(h,1,[]))';
                X = kron(eye(M), x);
                Y = X*H+W;
                r = zeros(1, M*(Q(Q_index)+K-1));
                r(1) = 1;
                for u = 1:length(r)
                   if mod(u, Q(Q_index)+1) == 0
                      r(u) = rho;
                     end
                end

                crb = (X'*C^-1*X)^-1;
                H_EST = crb*X'*C^-1*Y;
                MSE_h(Q_index, a , s, p) = mean((H_EST - H).^2,"all");

                value =zeros(M*N,1);
                for i = 1:length(value)
                    value(i) =  crb(i,i);
                end
                CRB(Q_index, s, p) = mean(value);
            end%MC
        end
    end
end
MSE_h = mean(MSE_h, 4);
CRB=mean(CRB, 3);

%alpha value
alph=3;%0    0.2475    0.4950    0.7425    0.9900
figure;grid on; box on;
%loglog(SNR, reshape(MSE_h(:,alph,:),[length(Q),length(SNR)]))
plot(SNR_dB, 10*log10(reshape(MSE_h(:,alph,:),[length(Q),length(SNR)])))
hold on;
%loglog(SNR, CRB,'--')
plot(SNR_dB, 10*log10(CRB),'--')
title('1.2(i)MSE_h vs SNR',['\alpha =',num2str(alpha(alph))] )
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(floor(Q)));
title(lgd,'Q')


%Q value
Que=3;%4    15    27    38    50
figure;grid on; box on;
%loglog(SNR, reshape(MSE_h(Que,:,:),[length(alpha),length(SNR)]))
plot(SNR_dB, 10*log10(reshape(MSE_h(Que,:,:),[length(alpha),length(SNR)])))
hold on;
%loglog(SNR, CRB(Que,:),'--')
plot(SNR_dB, 10*log10(CRB(Que,:)), '--')
title('1.2(i)MSE_h vs SNR',['Q =',num2str(floor(Q(Que)))] )
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(alpha));
title(lgd,'\alpha')
```

```
%% 1.2(ii) MIMO estimation
%clear all; clc;
disp('===1.2(ii)===')
M = 4;
N = 4;
K =4;
beta = [0.9, 0.5, 0.1];
SNR_dB = -10:2:30; %dB
SNR = 10.^(SNR_dB/10);%SNR_linear
alpha = linspace(0,0.99,5);
Q = floor(linspace(14, 50, 5));%Q_min>=K(M-1)+1
rho = 0.1;
Nmc = 1000;
MSE_h = zeros(length(Q), length(beta), length(alpha),length(SNR), Nmc);
CRB = zeros (length(Q), length(SNR), Nmc);

for Q_index = 1:length(Q)
    disp( ['Q =', num2str(Q(Q_index)), '/50'])
    for b =1 :length(beta)
        for a = 1:length(alpha)%different alpha
            for l= 1:N

                for i = 1:M
                    for k = 1:K
                        h(i,l,k) = alpha(a)^abs(i-l)*beta(b)^k;
                    end
                end
            end

            for s = 1:length(SNR)
                for p =1 :Nmc%MC
                    x = randn(N, Q(Q_index));
                    C = Cov_True(rho, M)/SNR(s);
                    w = chol(C, 'lower')* randn(M, Q(Q_index)+K-1);

                    X1 = convmtx(x(1, :)',K);
                    X2 = convmtx(x(2, :)',K);
                    X3 = convmtx(x(3, :)',K);
                    X4 = convmtx(x(4, :)',K);
                    XX = [X1,X2,X3,X4];
                    X = kron(eye(M), XX);
                    W = (reshape(w',1,[]))';

                    H = zeros(K*N*M,1);
                    f = 1;
                    for i=1:M
                        for l =1:N
                            for k =1:K
                                H(f) = h(i,l,k);
                                f=f+1;
                            end
                        end
                    end

                    Y = X*H+W;

                    r = zeros(1, M*(Q(Q_index)+K-1));
                    r(1) = 1;
                    for u = 1:length(r)
                        if mod(u, Q(Q_index)+1) == 0
                            r(u) = rho;
                        end
                    end

                    C = toeplitz(r)/SNR(s);
                    crb = (X'*C^-1*X)^-1;
                    H_EST = crb*X'*C^-1*Y;
                    MSE_h(Q_index, b, a , s, p) = mean((H_EST - H).^2,"all");

                    value =zeros(K*M*N,1);
                    for i = 1:length(value)
                        value(i) =  crb(i,i);
                    end
                    CRB(Q_index, s, p) = mean(value);
                end%MC
            end
        end
    end
end

MSE_h = mean(MSE_h, 5);
CRB=mean(CRB, 3);

%alpha value
alph=3;%0    0.2475    0.4950    0.7425    0.9900
bet1 = 1;%[0.9, 0.5, 0.1]
figure;grid on; box on;
%loglog(SNR, reshape(MSE_h(:,bet1, alph,:),[length(Q), length(SNR)]))
plot(SNR_dB, 10*log10(reshape(MSE_h(:,bet1, alph,:),[length(Q),
length(SNR)])))
hold on;
%loglog(SNR, CRB,'--')
plot(SNR_dB, 10*log10(CRB), '--')
title('1.2(ii) MSE_h vs SNR',['\alpha =',num2str(alpha(alph)), '\beta =',
num2str(beta(bet1))] )
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(Q));
title(lgd,'Q')

%Q value
Que=3;%4    15    27    38    50
bet2 = 1;%[0.9, 0.5, 0.1]
figure;grid on; box on;
%loglog(SNR, reshape(MSE_h(Que,bet2,:,:), [length(alpha),length(SNR)]))
plot(SNR_dB, 10*log10(reshape(MSE_h(Que,bet2,:,:),
[length(alpha),length(SNR)])))
hold on;
%loglog(SNR, CRB(Que,:),'--')
plot(SNR_dB, 10*log10(CRB(Que,:)), '--')
title('1.2(ii) MSE_h vs SNR',['Q =',num2str(Q(Que)), '\beta
=',num2str(beta(bet2))] )
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(alpha));
title(lgd,'\alpha')

%beta value
Que=4;%4    15    27    38    50
alph=3;%0    0.2475    0.4950    0.7425    0.9900
figure;grid on; box on;
%loglog(SNR, reshape(MSE_h(Que,:,alph,:),[length(beta), length(SNR)]))
plot(SNR_dB, 10*log10(reshape(MSE_h(Que,:,alph,:),[length(beta),
length(SNR)])))
hold on;
%loglog(SNR, CRB(Que,:),'--')
plot(SNR_dB, 10*log10(CRB(Que,:)),'--')
title('1.2(ii) MSE_h vs SNR',['Q =',num2str(Q(Que)), '\alpha
=',num2str(alpha(alph))] )
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(beta));
title(lgd,'\beta')
```



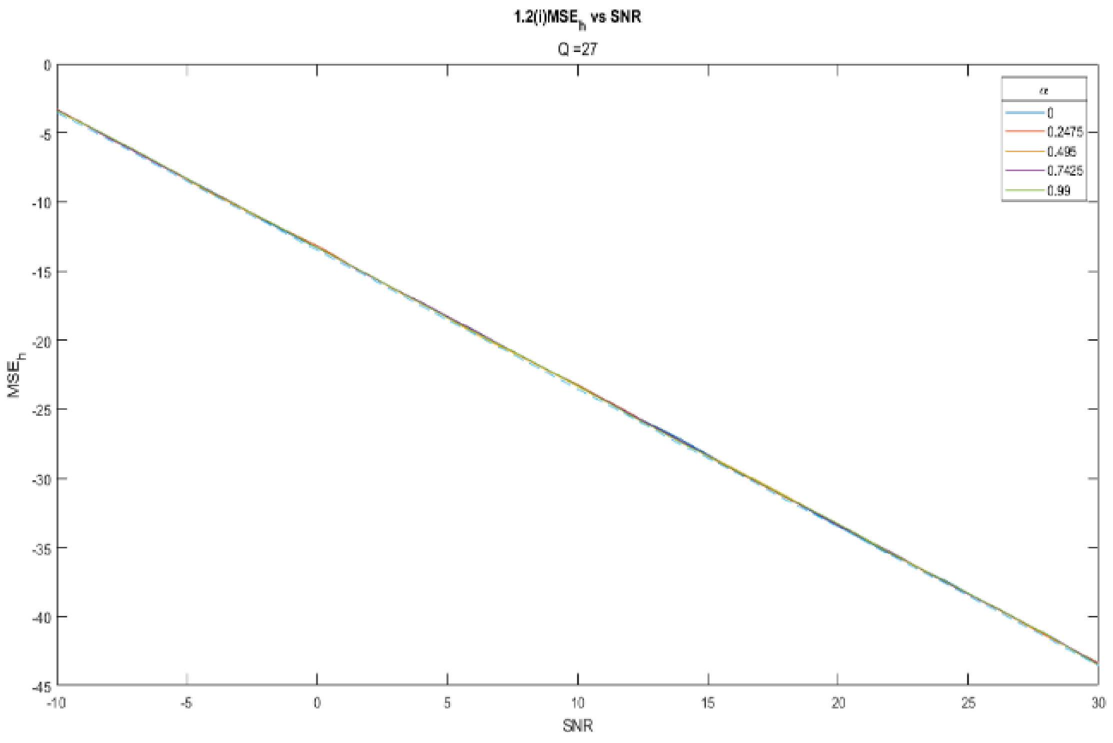
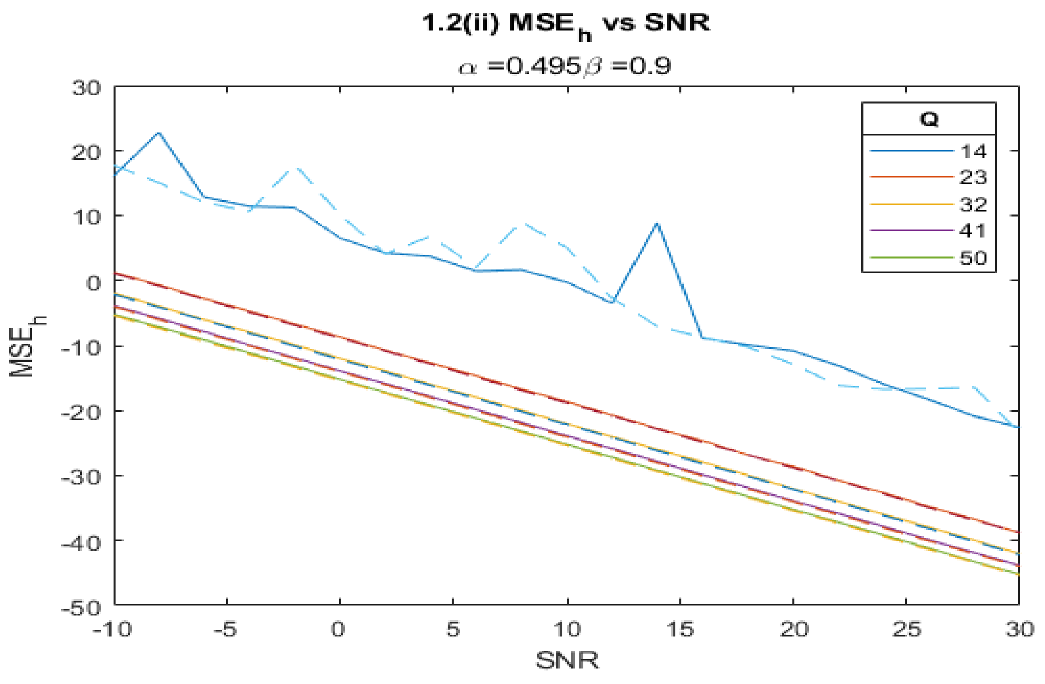
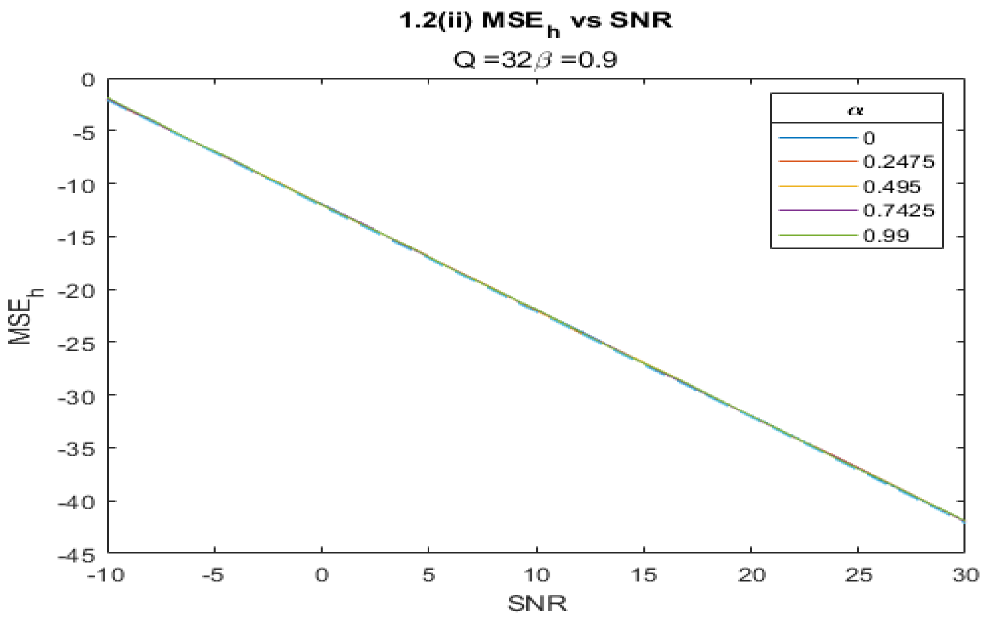


Fig 1.2i.1 and 1.2i.2 Dependency of MSE and CRB on SNR for different Q; dependency of MSE on signal SNR for different α values for a memoryless filter





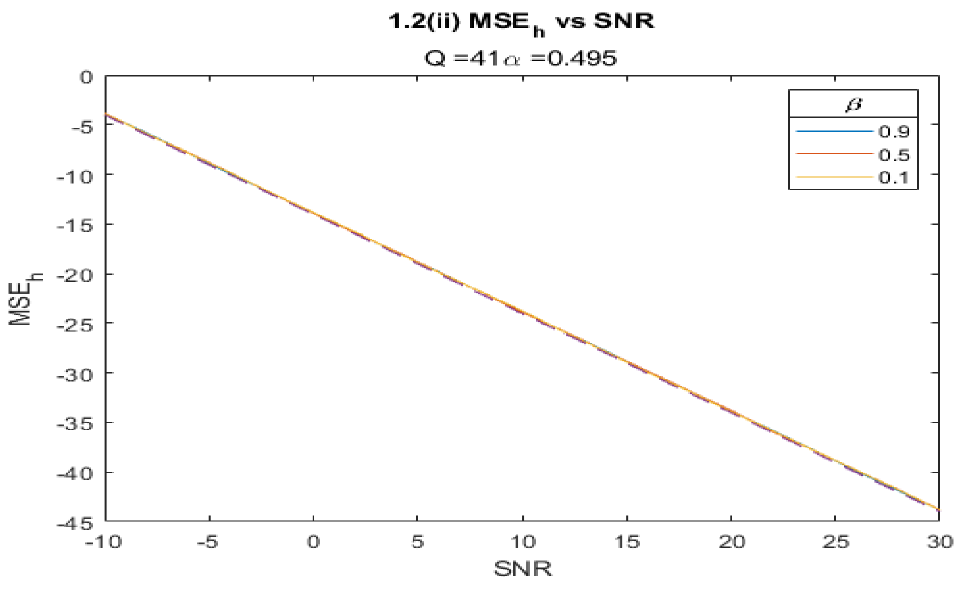
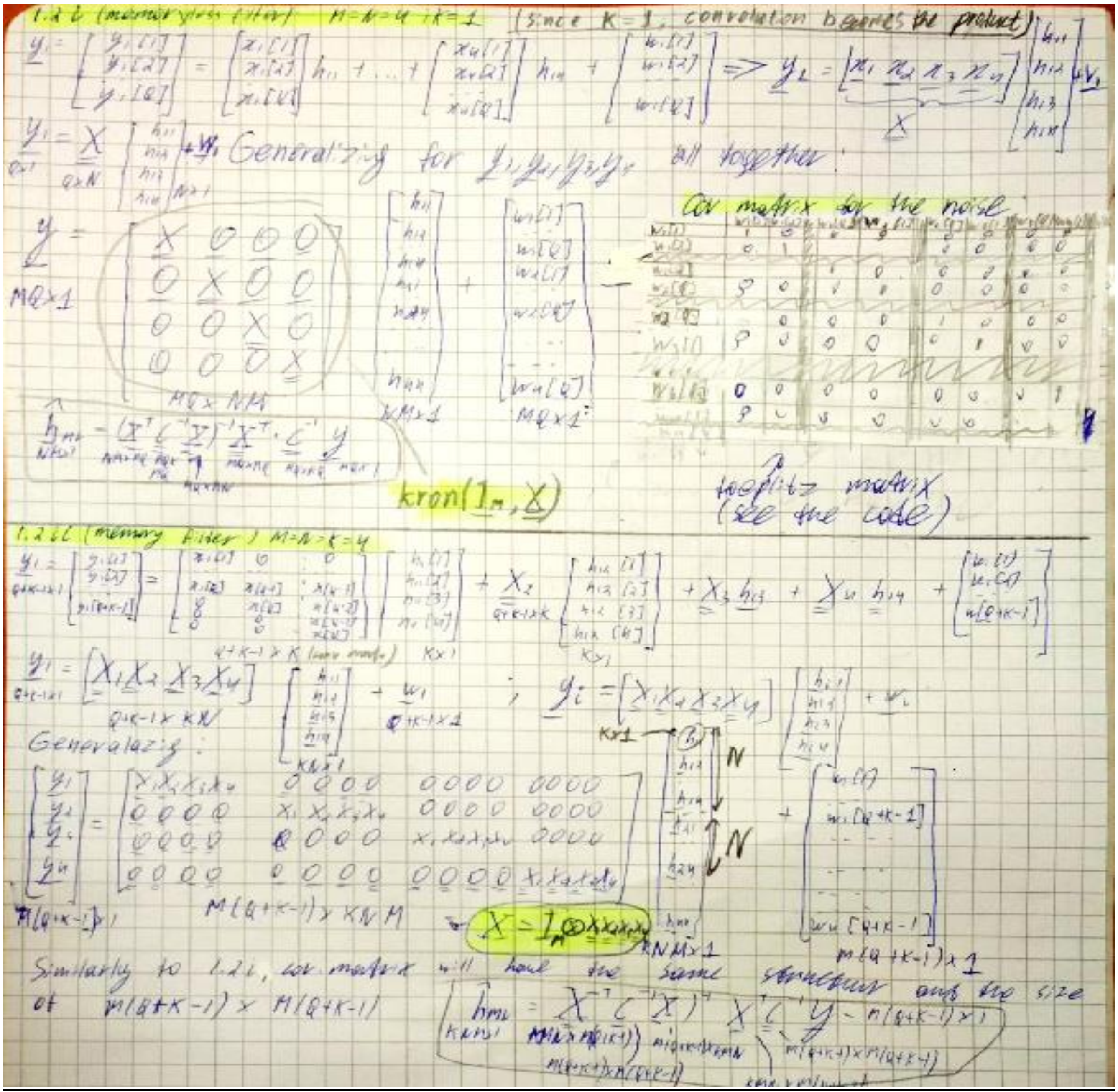


Fig 1.2ii.1, 1.2ii.2 and 1.2.ii.3 Dependency of MSE and CRB on SNR for different Q; dependency of MSE on signal SNR for different α values; dependency of MSE on SNR for different β values for a memory filter



1.2 b (memoryless filter) $M=N=4$, $K=1$ (since $K=1$, convolution becomes the product)

$$\underline{y}_1 = \begin{bmatrix} y_1[1] \\ y_1[2] \\ \vdots \\ y_1[Q] \end{bmatrix} = \begin{bmatrix} x_1[1] \\ x_1[2] \\ \vdots \\ x_1[Q] \end{bmatrix} h_{11} + \dots + \begin{bmatrix} x_4[1] \\ x_4[2] \\ \vdots \\ x_4[Q] \end{bmatrix} h_{14} + \begin{bmatrix} w_1[1] \\ w_1[2] \\ \vdots \\ w_1[Q] \end{bmatrix} \Rightarrow \underline{y}_1 = \underbrace{[\underline{x}_1\ \underline{x}_2\ \underline{x}_3\ \underline{x}_4]}_{\underline{X}} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{14} \end{bmatrix} + \underline{w}_1$$

$$\underset{Q\times 1}{\underline{y}_1} = \underset{Q\times N}{\underline{X}} \underset{N\times 1}{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{14} \end{bmatrix}} + \underline{w}_1$$ Generalizing for y_1, y_2, y_3, y_4 all together:

$$\underset{MQ\times 1}{\underline{y}} = \underset{MQ\times NM}{\begin{bmatrix} \underline{X} & 0 & 0 & 0 \\ 0 & \underline{X} & 0 & 0 \\ 0 & 0 & \underline{X} & 0 \\ 0 & 0 & 0 & \underline{X} \end{bmatrix}} \underset{NM\times 1}{\begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{14} \\ h_{21} \\ \vdots \\ h_{24} \\ \vdots \\ h_{44} \end{bmatrix}} + \underset{MQ\times 1}{\begin{bmatrix} w_1[1] \\ \vdots \\ w_1[Q] \\ w_2[1] \\ \vdots \\ w_2[Q] \\ \vdots \\ w_4[Q] \end{bmatrix}}$$

kron$(\underline{I}_M, \underline{X})$

$$\underset{NM\times 1}{\hat{\underline{h}}_{ML}} = (\underline{X}^T \underline{C}^{-1} \underline{X})^{-1} \underline{X}^T \underline{C}^{-1} \underline{y}$$

Cov matrix for the noise

	$w_1[1]$	$w_1[2]$	…	…	…	…	…	…
$w_1[1]$	1	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
$w_1[2]$	0	1	[illegible]	[illegible]	0	0	0	0
$w_2[1]$	[illegible]	[illegible]	1	0	0	0	0	0
$w_2[Q]$	0	0	1	1	0	0	0	0
$w_3[Q]$	[illegible]	0	0	0	1	0	0	0
$w_3[1]$	0	0	0	0	0	1	0	0
$w_4[Q]$	0	0	0	0	0	0	0	1
[illegible]	0	0	0	0	0	0	[illegible]	[illegible]

toeplitz matrix (see the code)

1.2 b C (memory filter) $M=N=K=4$

$$\underset{Q+K-1\times 1}{\underline{y}_1} = \begin{bmatrix} y_1[1] \\ y_1[2] \\ \vdots \\ y_1[Q+K-1] \end{bmatrix} = \underset{Q+K-1\times K\ (\text{toep. matrix})}{\begin{bmatrix} x_1[1] & 0 & 0 \\ x_1[2] & x_1[1] & \vdots \\ \vdots & x_1[Q] & x_1[K-1] \\ 0 & 0 & x_1[Q] \end{bmatrix}} \underset{K\times 1}{\begin{bmatrix} h_{11}[1] \\ h_{11}[2] \\ h_{11}[3] \\ h_{11}[4] \end{bmatrix}} + \underset{Q+K-1\times K}{\underline{X}_2} \underset{K\times 1}{\begin{bmatrix} h_{12}[1] \\ h_{12}[2] \\ h_{12}[3] \\ h_{12}[4] \end{bmatrix}} + \underline{X}_3 \underline{h}_{13} + \underline{X}_4 \underline{h}_{14} + \begin{bmatrix} w_1[1] \\ w_1[2] \\ \vdots \\ w_1[Q+K-1] \end{bmatrix}$$

$$\underset{Q+K-1\times 1}{\underline{y}_1} = \underset{Q+K-1\times KN}{[\underline{X}_1 \underline{X}_2 \underline{X}_3 \underline{X}_4]} \underset{KN\times 1}{\begin{bmatrix} \underline{h}_{11} \\ \underline{h}_{12} \\ \underline{h}_{13} \\ \underline{h}_{14} \end{bmatrix}} + \underset{Q+K-1\times 1}{\underline{w}_1} \quad ; \quad \underline{y}_i = [\underline{X}_1 \underline{X}_2 \underline{X}_3 \underline{X}_4] \begin{bmatrix} \underline{h}_{i1} \\ \underline{h}_{i2} \\ \underline{h}_{i3} \\ \underline{h}_{i4} \end{bmatrix} + \underline{w}_i$$

Generalazing:

$$\underset{M(Q+K-1)\times 1}{\begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \\ \underline{y}_3 \\ \underline{y}_4 \end{bmatrix}} = \underset{M(Q+K-1)\times KNM}{\begin{bmatrix} X_1X_2X_3X_4 & 0000 & 0000 & 0000 \\ 0000 & X_1X_2X_3X_4 & 0000 & 0000 \\ 0000 & 0000 & X_1X_2X_3X_4 & 0000 \\ 0000 & 0000 & 0000 & X_1X_2X_3X_4 \end{bmatrix}} \underset{KNM\times 1}{\begin{bmatrix} \underline{h}_{11} \\ \underline{h}_{12} \\ \vdots \\ \underline{h}_{14} \\ \underline{h}_{21} \\ \vdots \\ \underline{h}_{24} \\ \vdots \\ \underline{h}_{44} \end{bmatrix}} + \underset{M(Q+K-1)\times 1}{\begin{bmatrix} w_1[1] \\ \vdots \\ w_1[Q+K-1] \\ \vdots \\ w_4[Q+K-1] \end{bmatrix}}$$

($\underline{h}_{11}$: $K\times 1$; $\underline{h}_{11}\ldots\underline{h}_{14}$: N; $\underline{h}_{21}\ldots\underline{h}_{24}$: N)

$\rightarrow \boxed{\underline{X} = \underline{I}_M \otimes X_1X_2X_3X_4}$

Similarly to 1.2 i, cov matrix will have the same structure and the size of $M(Q+K-1)\times M(Q+K-1)$

$$\underset{KNM\times 1}{\hat{\underline{h}}_{ML}} = (\underline{X}^T \underline{C}^{-1} \underline{X})^{-1} \underline{X}^T \underline{C}^{-1} \underline{y} - M(Q+K-1)\times 1$$

$M(Q+K-1)\times M(Q+K-1)$; $M(Q+K-1)\times KNM$

-*Sect.3 Conclusion*

It is seen that MSE of the channel (which composes the variance plus the squared bias), gets lower as the Q value and(or) as the SNR increases. In addition, our estimate is nealy approaches the CRB, which is basically the best estimate possible, coinsiding with the theory, since the MSE should idealy be on the CRB line (asympt. efficient).

1.3 MIMO deconvolution

-Sect.1 Theoretical tools and methods adopted to solve the problem

For this problem MLE was used in order to estimate the system. After that, the received signal was estimated using ML and MMSE methods. In order to estimate the overall performance and the tradeoff between the Q and the MSE, special metric was plotted. Additionally, Kronoker product and Monte Carlo simulation were applied.

-Sect.2 Solution

```matlab
%% 1.3 MIMO deconvolution
disp('===1.3===')
M = 4;
N = 4;
SNR_dB = -10:2:30; %dB
P = 200;
SNR = 10.^(SNR_dB/10);%SNR_linear
Q = floor(linspace(5, P-1, 20));%Q_min>=K(M-1)+1
rho = 0.1;
alpha = 0.5;
Nmc = 1000;%10^3;
h = zeros(M, N);
MSE_h= zeros(length(SNR), length(Q), Nmc);
MSE_x_ML= zeros(length(SNR), length(Q), Nmc);
MSE_x_MMSE= zeros(length(SNR), length(Q), Nmc);

for l= 1:N
    for i = 1:M
        h(i,l) = alpha^abs(i-l);
    end
end

for Q_index = 1:length(Q)
    disp( ['Q =', num2str(Q(Q_index)), '/199'])
    for s = 1:length(SNR)
        for p =1 :Nmc%MC
            x =  randn(floor(Q(Q_index)),N);
            c = Cov_True(rho, M)/ SNR(s);%4X4
            w = chol(c, 'lower')* randn(M, floor(Q(Q_index)));

            W = (reshape(w',1,[]))';
            H = (reshape(h,1,[]))';
            X = kron(eye(M), x);
            Y = X*H+W;

            %C = kron(eye(floor(Q(Q_index))), c);
            r = zeros(1, M*(Q(Q_index)+K-1));
            r(1) = 1;
            for u = 1:length(r)
                if mod(u, Q(Q_index)+1) == 0
                    r(u) = rho;
                end
            end
            C = toeplitz(r)/SNR(s);
            crb = (X'*C^-1*X)^-1;

            H_EST = (X'*C^-1*X)^-1*X'*C^-1*Y;
            MSE_h(s, Q_index, p) = mean((H_EST - H).^2,"all");

            x2 =  randn(N, floor(P-Q(Q_index)));
            w2 = chol(c, 'lower')* randn(M, floor(P-Q(Q_index)));
            y2 = h * x2 + w2;
            H_ml = reshape(H_EST, M, N);
            X_EST_ML = (H_ml' * c^-1 * H_ml) \ H_ml' * c^-1 * y2;
            X_EST_MMSE = (H_ml' * c^-1 * H_ml + eye(N)) \ H_ml' * c^-1 * y2;

            MSE_x_ML(s,Q_index, p) = mean((x2(:) - X_EST_ML(:)).^2);
            MSE_x_MMSE(s,Q_index, p) = mean((x2(:) - X_EST_MMSE(:)).^2);
        end%MC
    end
end

 MSE_h = mean(MSE_h, 3);
 MSE_x_ML= mean(MSE_x_ML, 3);
 MSE_x_MMSE= mean(MSE_x_MMSE, 3);


figure;
%loglog(SNR, MSE_h);
plot(SNR_dB, 10*log10(MSE_h))
title('1.3 MSE_h vs SNR')
xlabel('SNR')
ylabel('MSE_h')
lgd = legend(string(floor(Q)));
title(lgd,'Q')


figure;
%loglog(SNR, MSE_x_ML);
plot(SNR_dB, 10*log10(MSE_x_ML))
title('1.3 MSE_x(ML) vs SNR')
xlabel('SNR')
ylabel('MSE_x')
lgd = legend(string(Q));
title(lgd,'Q')

figure;
%loglog(SNR, MSE_x_MMSE);
plot(SNR_dB, 10*log10(MSE_x_MMSE))
title('1.3 MSE_x(MMSE) vs SNR')
xlabel('SNR')
ylabel('MSE_x')
lgd = legend(string(Q));
title(lgd,'Q')

metric_ml = zeros(length(Q), length(SNR));
metric_mmse = zeros(length(Q), length(SNR));
for Q_index = 1:length(Q)
    for s = 1:length(SNR)
        metric_ml(Q_index, s) = (1-Q(Q_index)/P)*log(1+(SNR(s)./MSE_x_ML(s,
Q_index)));
        metric_mmse(Q_index, s) = (1-Q(Q_index)/P)*log(1+(SNR(s)./MSE_x_MMSE(s,
Q_index)));
    end
end

figure;
plot(Q, metric_ml)
title('1.3 Metric vs Q')
xlabel('Q')
ylabel('Metric')
lgd = legend(string(SNR_dB));
title(lgd,'SNR')
```

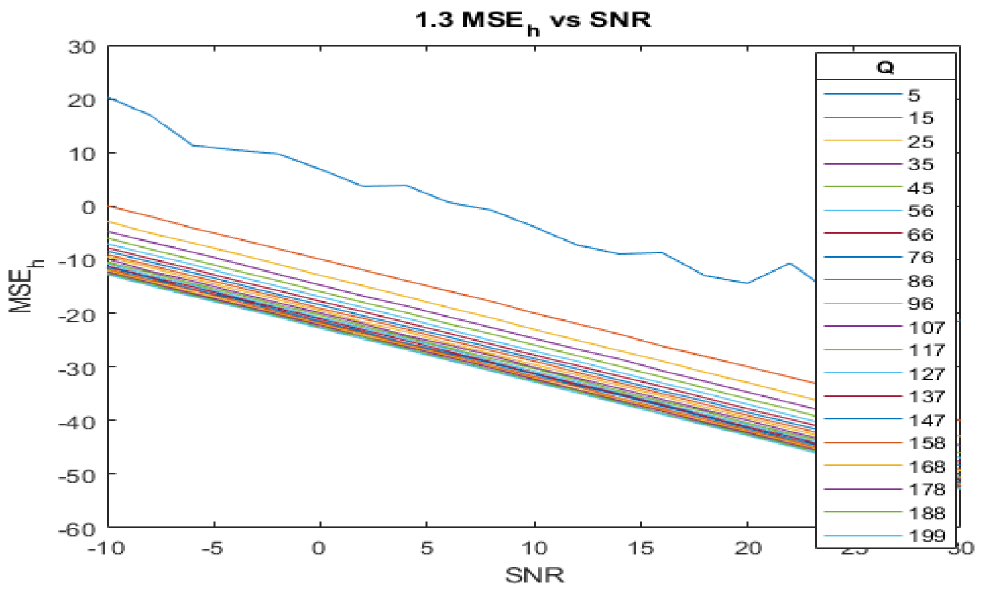


Fig 1.3.1Dependency of MSE of the channel on SNR

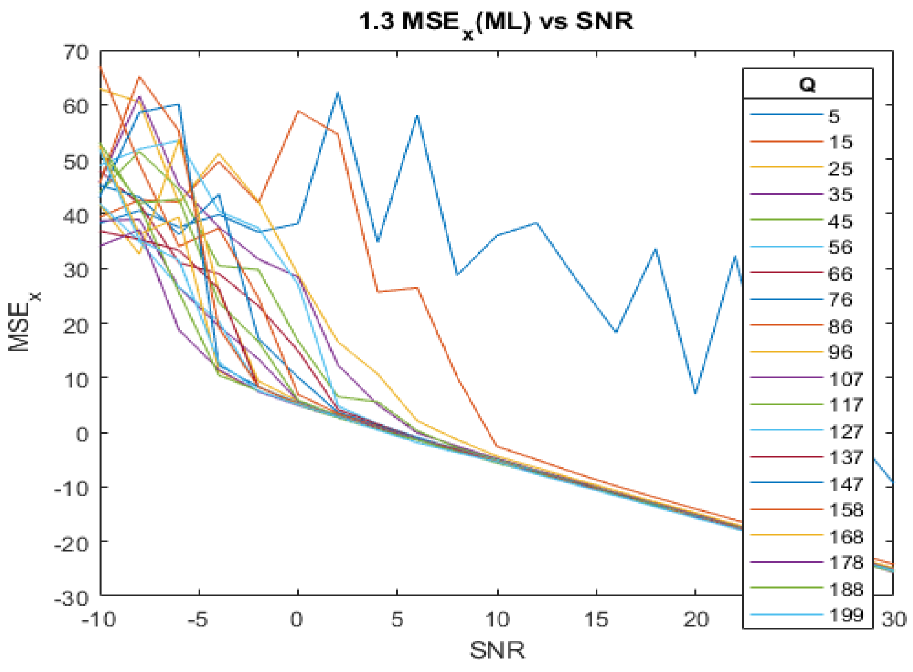


Fig 1.3.2 Dependency of MSE of the signal on SNR for ML

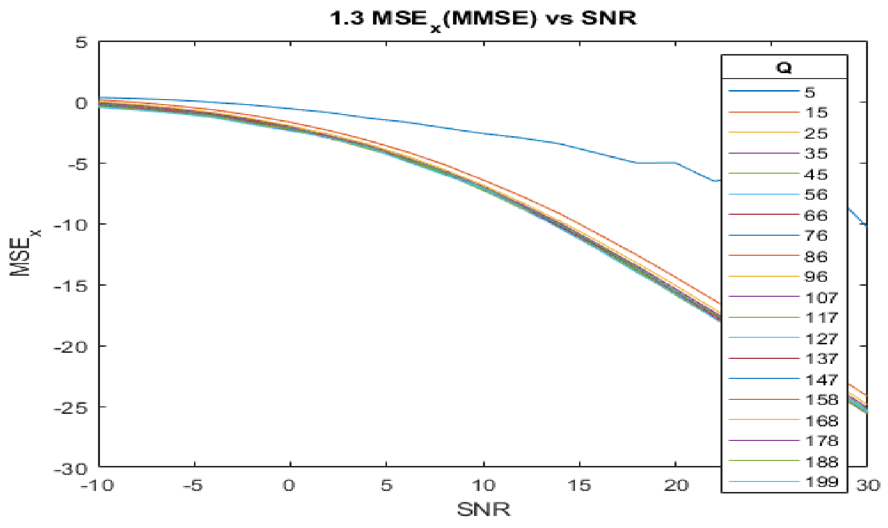


Fig 1.3.3 Dependency of MSE of the signal on SNR for MMSE

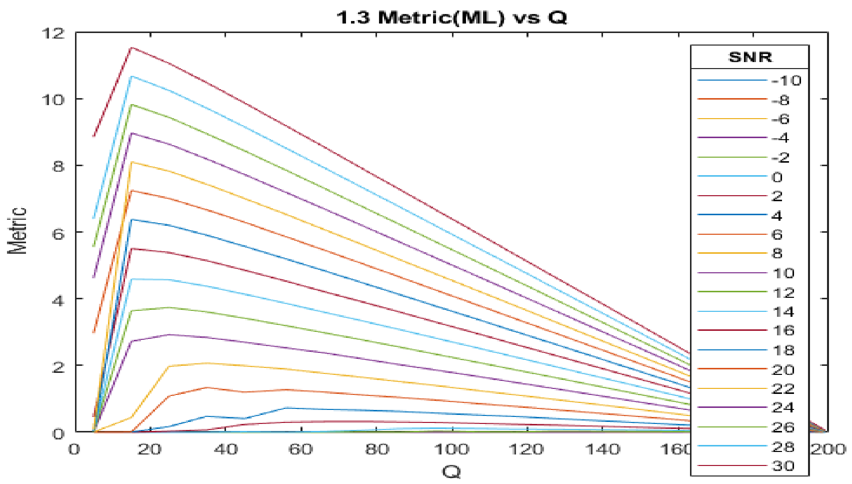
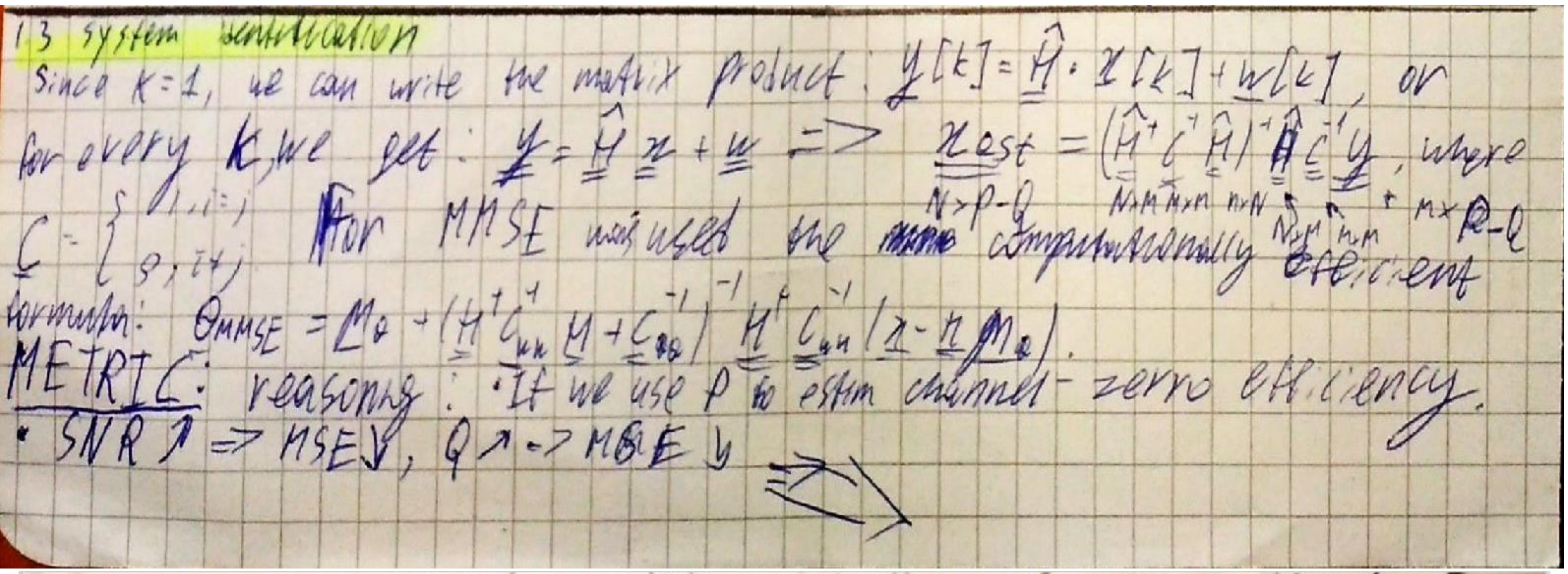


Fig 1.3.$\left(1-\frac{Q}{P}\right)\log(1+\frac{1}{MSE}SNR)$ vs Q for different SNR values

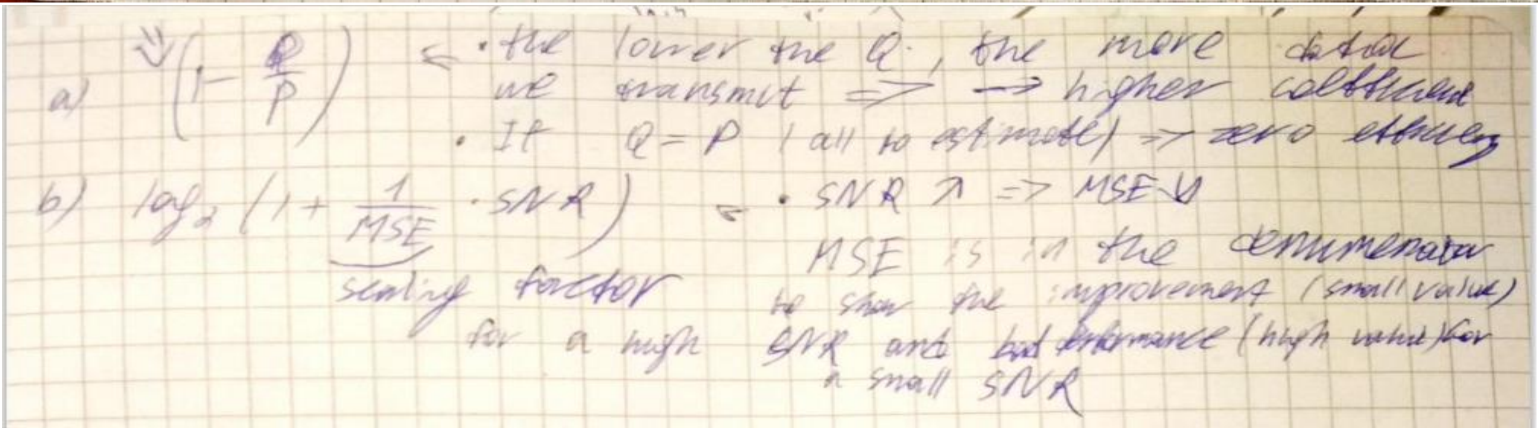

1.3 system [illegible]

Since $K=1$, we can write the matrix product: $\underline{y}[k] = \underline{\underline{\hat{H}}} \cdot \underline{x}[k] + \underline{w}[k]$, or for every k, we get: $\underline{y} = \underline{\underline{\hat{H}}}\,\underline{x} + \underline{w} \Rightarrow \underset{N\times P-Q}{\underline{x}_{est}} = (\underset{N\times M}{\hat{H}^T}\,\underset{M\times M}{C^{-1}}\,\underset{M\times N}{\hat{H}})^{-1}\,\underset{N\times M}{\hat{H}^T}\,\underset{M\times M}{C^{-1}}\,\underset{M\times P-Q}{\underline{y}}$, where

$\underline{C} = \begin{cases} 1, i=j \\ \rho, i\neq j \end{cases}$ For MMSE was used the ~~[illegible]~~ computationally efficient formula: $\theta_{MMSE} = \mu_\theta + (\underline{\underline{H}}^T C_{ww}^{-1} \underline{H} + \underline{\underline{C}}_{\theta\theta}^{-1})^{-1} \underline{\underline{H}}^T \underline{\underline{C}}_{ww}^{-1} (\underline{x} - \underline{H}\,\mu_\theta)$.

METRIC: reasoning: • If we use P to estim channel - zerro efficiency.

• SNR ↗ ⇒ MSE ↘, Q ↗ ⇒ MSE ↘ ⇒


a) $\Rightarrow \left(1 - \frac{Q}{P}\right)$ ⇐ • the lower the Q, the more data we transmit ⇒ → higher [illegible]

• If $Q = P$ (all to estimate) ⇒ zero efficiency

b) $\log_2\left(1 + \underbrace{\frac{1}{MSE}}_{\text{scaling factor}} \cdot SNR\right)$ ⇐ • SNR ↗ ⇒ MSE ↘

MSE is in the denominator to show the improvement (small value) for a high SNR and bad performance (high value) for a small SNR

Thus, the metric used for the evaluation is $\left(1 - \frac{Q}{P}\right)\log(1 + \frac{1}{MSE}SNR)$

-Sect.3 Conclusion

Comparing ML and MMSE for signal estimation, we can conclude that is is better to use MMSE for the small values of SNR(coincides with a-priori). As the power of the signal increases, the behaviour of is approximately identical for ML and MMSE. From the metric intridused in this section we can witness, that the maximum (traidoff) drifts towords right for low SNR values, meaning that for high values of SNR less sampes are required for channel estimation, thus more data can be transmittted.

Homework #2, AA2021-2022: Tracking and spatial filtering

1.1 Single source tracking

-Sect.1 Theoretical tools and methods adopted to solve the problem

For this problem beamforming was applied for estimation of the DoA, which corresponds to the maximum gain. In order to track the angle, Kalman filter was applied (extended, since the observation is a non-linear function of the state).

-Sect.2 Solution

```
%% hw2 Array Processing
%% 1)single source tracking
[N, L, P] = size(y);
SNR_dB = 0:2:40;
SNR = 10.^(SNR_dB/10);
fo = 2.9e9;
c = 3*10^8;
d = c / (2*fo);
dl = 50;

K = L/dl;% number of steps considered

lo = c / fo;
wo = 2*pi/lo;
V = 2;
dt = 1;

aoa_true = rad2deg(atan2(track_s1(2,:), track_s1(1,:)));
%% See the trajectory
pos_a = [(0:N-1).'*d zeros(N,1) ];

figure; hold on;
plot(track_s1(1,:), track_s1(2,:), '-x');
plot(track_s1(1,1), track_s1(2,1), 'g-o');
plot(pos_a(:,1), pos_a(:,2), 'o');
%% EKF

A = [1 0 dt 0; ...%x, y, vx, vy
     0 1 0 dt; ...
     0 0 1 0; ...
     0 0 0 1];

C_alfa = [0 0 0 0; ...
          0 0 0 0; ...
          0 0 2*cos(pi/360) 0; ...
          0 0 0 2*sin(pi/360)] + abs(1e-3*diag(randn(4,1)));

B=@(x)[1/(1+(x(2)/(x(1)))^2) *(-x(2))/((x(1)^2)),...%derivatives of the atan wrt. The state components
       1/(1+(x(2)/(x(1)))^2) *(1/(x(1))),...
       0,...
       0];

az = 0: pi/1080 : pi;
b_c = exp(1i * wo * d * (0:N-1).' * cos(az)) / sqrt(N);

angle_pred = zeros(K,P);

beam_ang = zeros(1,K);
state_est = zeros(4,K, P);
for p = 1:P
    angle_pred(1,p) = rad2deg(atan2(track_s1(2), track_s1(1)));

    %initialization
    theta_pp = ([track_s1(1), track_s1(2), track_s1(3)*cos(track_s1(4))*dt, track_s1(3)*sin(track_s1(4))*dt])';
    P_pp = eye(4)* 10^3;

    for i = 2:K
        %Find the angle/observation
        samples = 1+(i-1)*dl:i*dl;
        [value, ind] = max(mean(abs(b_c' * y(:,samples, p)).^2, 2));

        %Prediction
        theta_cp = A * theta_pp;
        P_cp = A * P_pp * A' + C_alfa;

        % Update:
        G = P_cp*B(theta_cp)'*(B(theta_cp)*P_cp*B(theta_cp)'+1/SNR(p))^-1;
        b = atan2(theta_cp(2), theta_cp(1));

        angle_pred(i,p) = rad2deg(b);

        theta_cc = theta_cp + G*(az(ind)-b);
        P_cc = P_cp - G *B(theta_cp)*P_cp;

        theta_pp = theta_cc;
        P_pp = P_cc;
    end
end
%% MSE
MSE =zeros(P,1);
for p = 1:P
  MSE(p) = mean( (aoa_true' - angle_pred(:, p)).^2 );
end

figure;
plot(SNR_dB, 10*log10(MSE));
title('MSE vs \sigma^2_x')
xlabel('\sigma_X^2')
ylabel('MSE')
%% close all
figure;
plot(aoa_true,'-r');
grid on;
hold on
plot(angle_pred(:,1), '-g')
hold on
plot(angle_pred(:,21), '-b')
title('Angle vs Time')
xlabel('Time')
ylabel('Angle')
legend("True observation", "SNR(dB) = 0 dB", "SNR(dB) = 40 dB")
```

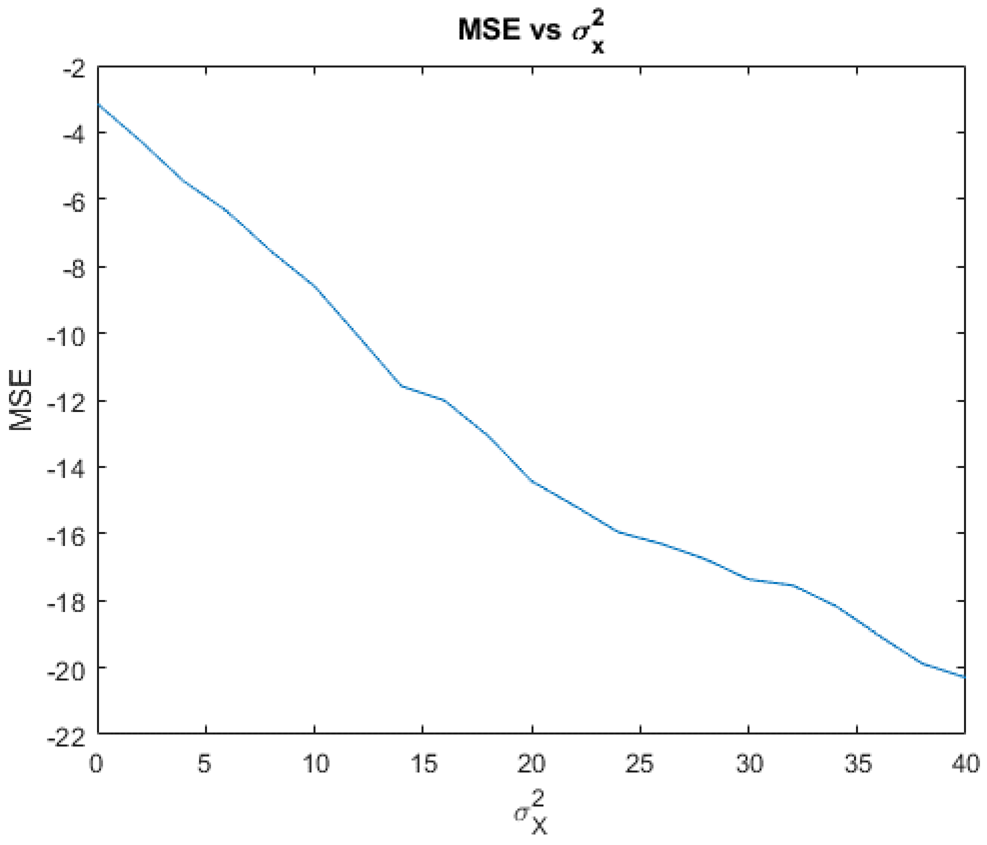


Fig 2.1.1 Dependency of MSE of the signal on SNR for the observed angle

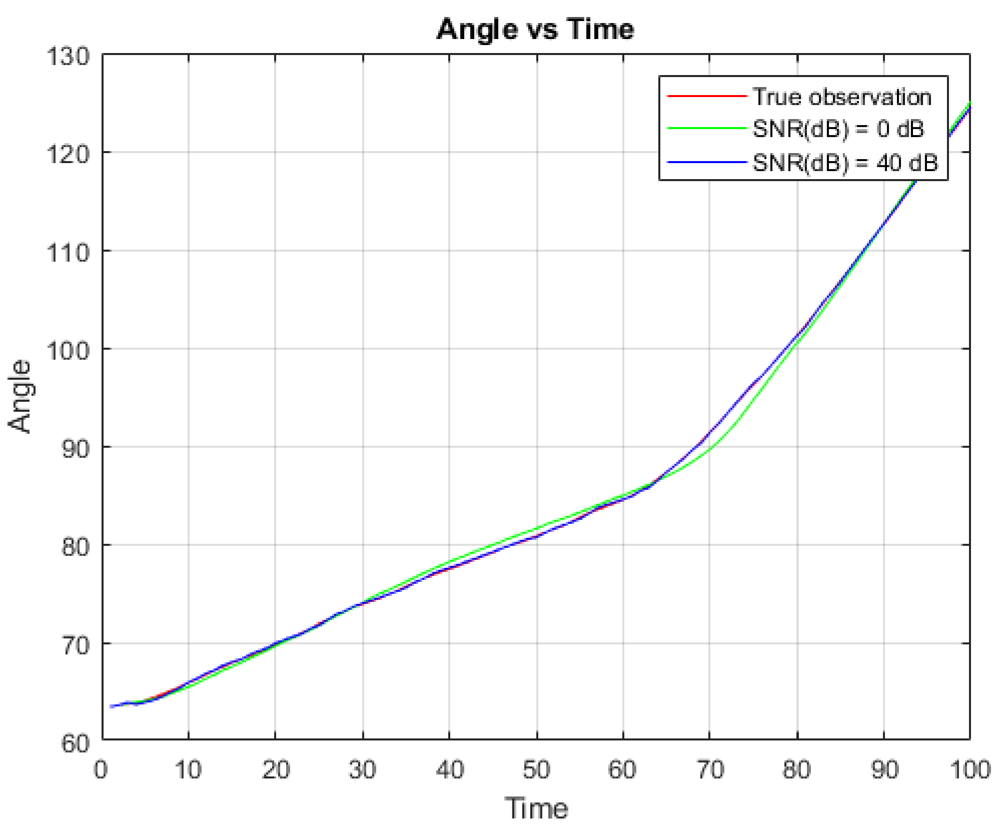


Fig 2.1.2 Tracked and true angles at different time instances

-Sect.3 Conclusion

In order to make the observation linearly dependent on the state, it is better for this case to use state vector [x, y, v_x, v_y], rather than [x, y, v, ψ]. From the last two plots we see that it the results are more accurate for a source with a larger SNR, rather than for the one with a smaller one.

1.2 Sources separation

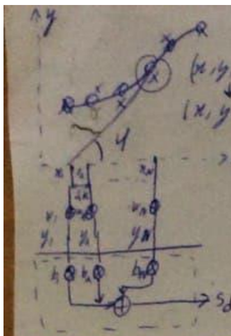
-Sect.1 Theoretical tools and methods adopted to solve the problem

For this problem beamforming coefficients were estimated to achieve the desired only the desired source. In contrast to the previous exercise, which shows a more “brute force” approach, in this part we don’t scan the entire field, but for computational efficiency only a small area. Using DoA as an observation, we apply EKF to predict the angle (around which we scan) and multiply the coefficient with the received signal.

-Sect.2 Solution

```
%% 1.2 Source separation
clc, clear, close all;
load('HW2.2.mat');
%%
[N, L, P] = size(y);
SNR_dB = 0:2:40;
SNR = 10.^(SNR_dB/10);
fo = 2.9e9;
c = 3*10^8;
d = c / (2*fo);
dl = 50;

K = L/dl;% number of steps considered

lo = c / fo;
wo = 2*pi/lo;
V = 2;
dt = 1;

%%
A = [1 0 dt 0; ...
     0 1 0 dt; ...
     0 0 1 0; ...
     0 0 0 1];
```

```matlab
C_alfa = [0 0 0 0; ...
          0 0 0 0; ...
          0 0 2*cos(pi/360) 0; ...
          0 0 0 2*sin(pi/360)];

B=@(x)[1/(1+(x(2)/(x(1)))^2) *(-x(2))/((x(1)^2)),...
       1/(1+(x(2)/(x(1)))^2) *(1/(x(1))),...
       0,...
       0];


%p = 21;
aoa_true = rad2deg(atan2(track_s2(2,:), track_s2(1,:)))';

s2_est = zeros(5000,P);


angle_pred = zeros(K,P);


for p = 1:P

    angle_pred(1,p) = rad2deg(atan2(track_s2(2), track_s2(1)));

    %scan everything for the first time
    scan_area = 0: pi/1080 : pi;
    b_init = exp(1i * wo * d * (0:N-1).' * cos(scan_area)) / sqrt(N);
    samples = 1:dl;
    g  = mean(abs(b_init' * y(:,samples, p)).^2, 2);
    [value, ind] = max(g);

    b2_init = exp(1i * wo * d * (0:N-1).' * cos(scan_area(ind))) / sqrt(N);
    s2_est(1:50,p) = (b2_init' * y(:,samples, p)).';


    %initialization
    theta_pp = ([track_s2(1), track_s2(2), track_s2(3)*cos(track_s2(4))*dt, track_s2(3)*sin(track_s2(4))*dt])';
    P_pp = eye(4)* 10^3;

    for i = 2:K

        %Prediction
        theta_cp = A * theta_pp;
        P_cp = A * P_pp * A' + C_alfa;

        b = atan2(theta_cp(2), theta_cp(1));%predicted angle

        angle_pred(i,p) = rad2deg(b);


        %scan around the predicted angle
        delta = 5*pi/180;
        scan_area = b - delta: pi/(20*360) : b + delta;

        %max in the area
        b_c = exp(1i * wo * d * (0:N-1).' * cos(scan_area)) / sqrt(N);
        samples = 1+(i-1)*dl:i*dl;
        g  = mean(abs(b_c' * y(:,samples, p)).^2, 2);
        [value, ind] = max(g);

        %find the [b(AoA)*y']
        b2 = exp(1i * wo * d * (0:N-1).' * cos(scan_area(ind))) / sqrt(N);
        s2_est(samples,p) = (b2' * y(:,samples, p)).';

        %obeservation
        x = scan_area(ind);

        % Update
        G = P_cp*B(theta_cp)'*(B(theta_cp)*P_cp*B(theta_cp)'+1/SNR(p))^-1;
        theta_cc = theta_cp + G*(x-b);
        P_cc = P_cp - G *B(theta_cp)*P_cp;

        theta_pp = theta_cc;
        P_pp = P_cc;
    end
end
%% MSE
MSE = zeros(P,1);
for p =1:P
    MSE(p) = mean((s2_est(:,p) - s2).^2);
end

figure
plot(SNR_dB, 10*log10(smooth(MSE)))
title('MSE vs \sigma^2_x')
xlabel('\sigma_X^2')
ylabel('MSE')
```



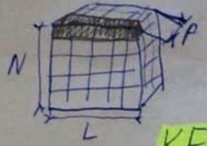
$\cdot\ \psi = \operatorname{atan}\left(\frac{x}{y}\right)$

$\psi = \operatorname{atan}\left(\frac{y}{x}\right)$ observation

$\Delta x = \frac{c}{2 f_0}$

$y = \mathbb{C}^{N \times L \times P}$ num. temp. samples, P_{SNR}

$N = 16$
$L = 5000$
$P = 21 \in [0:2:40]\,dB$



samples per step

$\delta\ell = 50$ ← to estimate the position

$L/\delta\ell$ ← number of steps

(I)

Extended Kalman Filter (for non-linear dynamic systems)

$\underline{\theta} = [x, y, v_x, v_y]$ ← state ← excitation

$$\begin{cases} \underline{\theta}[n] = \underline{a}(\underline{\theta}[n-1]) + \underline{\lambda}[n] & \leftarrow \text{state} \\ \underline{x}[n] = \underline{b}(\underline{\theta}[n]) + \underline{\beta}[n] & \leftarrow \text{observation} \end{cases}$$

← measurement model ← measurement noise

Initialization

$\underline{\hat{\theta}}[0|0] = \underline{\theta}[0]$ ← true

$\underline{P}[0|0] = \underline{I}_4 \cdot 10^3$

(I) Prediction

$\cdot\ \underline{\hat{\theta}}[n|n-1] = \underline{a}(\underline{\hat{\theta}}[n-1|n-1])$

$\cdot\ \underline{P}[n|n-1] = \underline{A}[n|n-1]\,\underline{P}[n-1|n-1]\,\underline{A}^T[n|n-1] + \underline{C}_{\lambda\lambda}$ a-priori

where

$$\underline{A}[n|n-1] = \left.\frac{\partial \underline{a}(\underline{\theta}[n-1])}{\partial \underline{\theta}[n-1]}\right|_{\underline{\theta}[n-1] = \underline{\hat{\theta}}[n-1|n-1]}$$

(II) Update/correction

$\cdot\ \underline{\hat{\theta}}[n|n] = \underline{\hat{\theta}}[n|n-1] + \underline{G}[n](\underline{x}[n] - \underline{b}(\underline{\hat{\theta}}[n|n-1]))$

where

$\underline{G}[n] = \underline{P}[n|n-1]\,\underline{B}^T[n|n-1]\left(\underline{B}[n|n-1]\,\underline{P}[n|n-1]\,\underline{B}^T[n|n-1] + \underline{C}_{\beta\beta}\right)^{-1}$ ← Kalman Gain

$$\underline{B}[n|n-1] = \left.\frac{\partial \underline{b}(\underline{\theta}[n])}{\partial \underline{\theta}[n]}\right|_{\theta[n] = \hat{\theta}[n|n-1]}$$

$\cdot\ \underline{P}[n|n] = \underline{P}[n|n-1] - \underline{G}[n]\,\underline{B}[n|n-1]\,\underline{P}[n|n-1]$

a-posteriori

KF (linear assumption)

$$\begin{cases} \underline{\theta}[n] = \underline{A}\,\underline{\theta}[n-1] + \underline{\lambda}[n] \\ \underline{x}[n] = \underline{B}\,\underline{\theta}[n] + \underline{\beta}[n] \end{cases}$$

(I) prediction

$\underset{4\times1}{\underline{\hat{\theta}}[n|n-1]} = \underset{4\times4}{\underline{A}}\,\underset{4\times1}{\underline{\hat{\theta}}[n-1|n-1]}$

$\underline{P}[n|n-1] = \underline{A}\,\underline{P}[n-1|n-1]\,\underline{A}^T + \underline{C}_{\lambda\lambda}$

(II) Update

$\underline{G}[n] = \underline{P}[n|n-1]\,\underline{B}^T(\underline{B}\,\underline{P}[n|n-1]\,\underline{B}^T + \underline{C}_{\beta\beta})^{-1}$

$\underline{\hat{\theta}}[n|n] = \underline{\hat{\theta}}[n|n-1] + \underline{G}[n](\underline{x}[n] - \underline{B}\,\underline{\hat{\theta}}[n|n-1])$

$\underline{\hat{P}}[n|n] = \underline{P}[n|n-1] - \underline{G}[n]\,\underline{B}\,\underline{P}[n|n-1]$

$\psi \sim N(0, \sigma_\psi^2)$

$$\underline{C}_{\lambda\lambda} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2 \end{bmatrix} + \lambda \xrightarrow{\text{sqrt}} \underline{\lambda}$$

$\sigma_{v_x}^2 = v\cos\sigma_\psi^2$

$\sigma_{v_y}^2 = v\sin\sigma_\psi^2$

$C_{\beta\beta} = \sigma_{\beta\beta}^2 = \frac{1}{SNR}$

In our case:

State changes linearly

$$\underbrace{\begin{bmatrix} x[n] \\ y[n] \\ v_x[n] \\ v_y[n] \end{bmatrix}}_{\underline{\theta}[n]} = \underbrace{\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\underline{A}} \underbrace{\begin{bmatrix} x[n-1] \\ y[n-1] \\ v_x[n-1] \\ v_y[n-1] \end{bmatrix}}_{\underline{\theta}[n-1]} + \underbrace{\begin{bmatrix} \lambda_x[n] \\ \lambda_y[n] \\ \lambda_{v_x}[n] \\ \lambda_{v_y}[n] \end{bmatrix}}_{\lambda[n]}$$

$\psi[n] = x[n] = \operatorname{atan}\left(\frac{y[n]}{x[n]}\right) + \beta[n]$

For EKF we have:

$$B = \left.\frac{\partial \underline{b}(\underline{\theta}[n])}{\partial \underline{\theta}[n]}\right|_{\theta[n] = \hat{\theta}[n|n-1]} = \begin{bmatrix} \frac{1}{1 + \frac{\hat{y}^2[n|n-1]}{\hat{x}^2[n|n-1]}} \cdot \left(-\frac{\hat{y}[n|n-1]}{\hat{x}^2[n|n-1]}\right) & \leftarrow \frac{\partial}{\partial x} \\ \frac{1}{1 + \frac{\hat{y}^2[n|n-1]}{\hat{x}^2[n|n-1]}} \cdot \left(\frac{1}{\hat{x}[n|n-1]}\right) & \leftarrow \frac{\partial}{\partial y} \\ 0 & \leftarrow \frac{\partial}{\partial v_x} \\ 0 & \leftarrow \frac{\partial}{\partial v_y} \end{bmatrix}^T$$

Problem 2 (Continued).

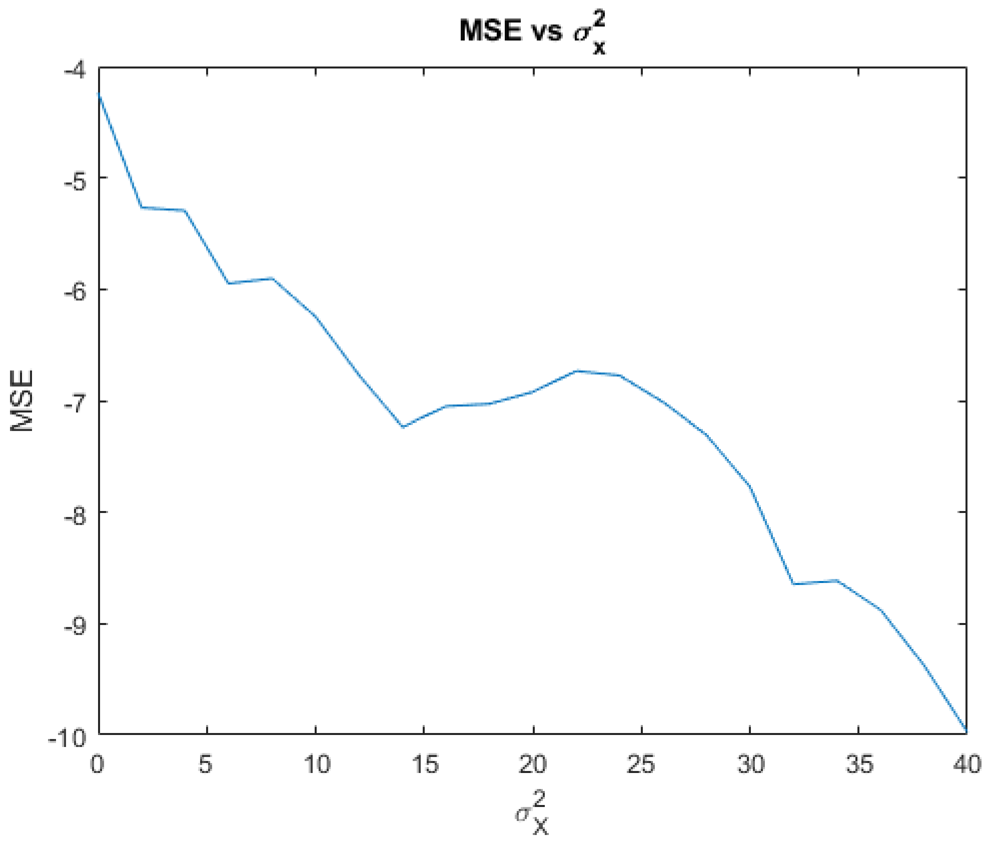


Fig 2.2.1 Dependency of the MSE of the estimated signal versus the power of the signal

-Sect.3 Conclusion

As the signals SNR increases, the MSE gets lower. In order to achieve a smoother curve, it would be better to have several realizations of the signal and then average using Monte-Carlo method. In addition, to the scanning procedure what could be proposed is to use a higher covariance of the noise at the point of intersection of two sources, which would mean that for this region me should trust the observation, since it can switch to the non-desired source.