```matlab
%% Tracking of a moving object which trajectory is disturbed by random acceleration

% Written by Irina Yareshko and Luca Breggion, Skoltech 2022

close all
clear
clc

set(0,'defaulttextInterpreter','latex');
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');

%% 1-2) Generate a true trajectory Xi of an object motion disturbed by normally distributed random ...
%       acceleration

%Creating of arrays
x = zeros(1, 200); %true data
V = zeros(1, 200); %velocity
Z = zeros(1, 200); %measurments

sigma2_n = 20^2;
n = randn*sqrt(sigma2_n); %random noise of measurments
%Initial data
N = 200;
T = 1;
x(1) = 5;
V(1) = 1;
Z(1) = x(1) + n; %the first measurment

%Generation of data
for i=2:N
    sigma2_a = 0.2^2;
    a = randn*sqrt(sigma2_a); %normally distributed random acceleration
    n = randn*sqrt(sigma2_n); %random noise of measurments
    V(i) = V(i - 1) + a*T;
    x(i) = x(i - 1) + V(i - 1)*T + a*T^2/2;
    Z(i) = x(i) + n;
end

%% 3-4) Presenting the system at state space and state vector estimations

%State matrixes
Fi = [1 T; 0 1];  %transition matrix
G  = [T/2; T];    %input matrix
H  = [1 0];       %observation matrix

% Kalman filter
X = zeros(2, 2, N + 1); %true data
P = zeros(2, 4, N + 1); %filtration error covariance matrix
```

```
K = zeros(2, N);
X_f = zeros(2, N);

%Initial conditions
X(:,:,1) = [2,0; 0,0];
P(:,1:2,1) = [10000 0; 0 10000];
Q = G*G.'*sigma2_a;
R = sigma2_n;

for i = 2:(N + 1)
    %Prediction of state vector at time i using i-1 measurements
    X(:,2,i-1) = Fi*X(:,1,i-1);
    X_f(:,i-1)  =Fi^7*X(:,1,i-1);

    %Prediction error covariance matrix
    P(:,3:4,i-1) = Fi*P(:,1:2,i-1)*Fi.'+Q;

    %Filter gain, weight of residual
    K(:,i-1) = P(:,3:4,i-1)*H.'*(H*P(:,3:4,i-1)*H.'+R)^(-1);

    %Improved estimate by incorporating a new measurement
    X(:,1 ,i) = X(:,2,i-1) + K(:,i-1)*(Z(i-1) - H*X(:,2,i-1));

    %Filtration error covariance matrix
    P(:,1:2,i) = (eye(2) - K(:,i-1)*H)*P(:,3:4,i-1);
end

x_Kalman(1:N) = X(1,1,2:N+1);

%% 5) Building of plots of true trajectory, measurments, filtered esttimates of
%     state vector

t = 1:N; %array of steps

figure(1)
plot(t,x,'c',t,Z,'m',t,x_Kalman,'k', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('True data','Measurments','Filtered Estimates of State Vector','FontSize',30);

%% 6) Filter fain and filtration error

figure(2)
plot(t, K(1,:),'r', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('Filter Gain K', 'FontSize', 30)
legend('Filter Gain', 'FontSize', 30)
```

```
ylim([0 1])

P_sq(1:N)=sqrt(P(1,1,1:N));

figure(3)
plot(t, P_sq,'r', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('Filtration error', 'FontSize', 30)
legend('Filtration error', 'FontSize', 30)

%% 7-8-9 + 10) Estimation of dinamics of mean-sqaured error of estimation over...
%     observation interval

clear all

%Initial data
m = 7;              % number of steps ahead
M = 500;            % number of runs
N = 200;            % observation interval
sigma2_n = 20^2;    % variance of noise
sigma2_a = 0.2^2;   % variance of accelerarion
T = 1;              % period of step

% Initialization
Error_X     = zeros(M, N);       % array of errors of filtered estimate
Error_X_f   = zeros(M, N);       % array of errors of forecasts
Error_X_f7  = zeros(M, N - 6);   % array of errors of forecasts
X_Kalman    = zeros(6, N, M);    % array of filtered data
x           = zeros(2, N);

for i = 1:M
    [x, Z] = data_gen(N,T,sigma2_n,sigma2_a);
    X_Kalman(:,:,i) = Kalman_filter(Z,T,m,sigma2_n,sigma2_a); % Kalman filter
    Error_X(i,:)    = (x(1,:) - X_Kalman(1,:,i)).^2; % errors of filtered estimate
    Error_X_f(i,:)  = (x(1,:) - X_Kalman(2,:,i)).^2; % errors of forecasts 1-step
    Error_X_f7(i,:) = (x(1,7:N) - X_Kalman(3,1:N-6,i)).^2; % errors of forecasts 7-step
end

%Final average value of Error over M runs
Final_Error_X    = sqrt(1/(M-1)*sum(Error_X));
Final_Error_X_f  = sqrt(1/(M-1)*sum(Error_X_f));
Final_Error_X_f7 = sqrt(1/(M-1)*sum(Error_X_f7));

%Building of plot of final errors of filtered estimate and errors of
%forecasts
t=1:(N-2);

%True error for filtration
```

```
figure(4)
plot(t,Final_Error_X(3:N),'m',t,X_Kalman(4,3:N,1),'k', 'LineWidth', 1.2);
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('Errors', 'FontSize', 30)
legend('True Estimation Error','Filtration Error Covariance Matrix', 'FontSize', 30)

t1 = 7:N;
figure(6)
plot(t, Final_Error_X(3:N),'m', t1, Final_Error_X_f7,'k', 'LineWidth', 1.2);
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('Errors', 'FontSize', 30)
legend('True estimation error','True error of 7-step prediction', 'FontSize', 30)

t = 1:N;
figure(8)
plot(t,X_Kalman(6,:,1), 'c','LineWidth', 1.2);
grid on; grid minor
xlabel('Step', 'FontSize', 30)
ylabel('K', 'FontSize', 30)
legend('Filter gain with $P^I$','FontSize', 30, 'interpreter', 'latex');

%% Point 11
N = 200;
T = 1;
X0 = [2; 0];
P0 = [10000 0; 0 10000];
sigma2_n = 20^2;
sigma2_a = 0.2^2;
m = 7;
M = 500;
err_filt = zeros(M, 1, N);

for i = 1:M
    [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
    [Z_filt, ~, P_mat] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
    err_filt(i,1,:) = (x - Z_filt).^2;
end

%% Pont 12
sigma2_a = 0;

for i = 1:M
    [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
    [Z_filt, ~, P_matr, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
    err_filt(i,1,:) = (x - Z_filt).^2;
end
fin_err_filt = sqrt( 1/(M-1) * sum(err_filt) );
```

```
sqrt_covm_errf = sqrt(P_matr(1,1,2:N + 1));

figure(121)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize', 30, 'location', 'best')
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

figure(122)
plot(1:N , K_arr(1,:),'g', 'LineWidth', 1.2)
grid on; grid minor
legend('Filter Gain', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Gain K', 'FontSize', 30)

figure(123)
plot(3:N, fin_err_filt(1,3:N),'m', 3:N, sqrt_covm_errf(1, 3:200),'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Estimation Error','Filtration Error Covariance Matrix', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Errors', 'FontSize', 30)

%% Pont 13

err_for = zeros(M, 1, N);

for i = 1:M
    [x, z] = data_gen(N, T, sigma2_n, 0.2^2); %0.2^2 = sigma2_a;
    [Z_filt, X_pred, ~, ~] = Kalman_Filter_2(z, m, 0, X0, P0); %Q = 0;
    err_filt(i,1,:) = (x - Z_filt).^2;
    err_for(i,1,:) = (x - X_pred).^2;
end

fin_err_filt = sqrt( 1/(M-1) * sum(err_filt) );
fin_err_for = sqrt( 1/(M-1) * sum(err_for) );

figure(131)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'b', 'LineWidth', 1.2)
grid on; grid minor
legend('True','Measurements','Filtered Estimates of State Vector', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

figure(132)
plot(3:N, fin_err_filt(1,3:N),'m', 3:N, fin_err_for(1,3:N),'k', 'LineWidth', 1.2)
grid on; grid minor
```

```
legend('Filtered Estimate Error', 'Prediction Estimate Error', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Errors', 'FontSize', 30)

figure(133)
plot(3:N, fin_err_filt(1,3:N),'m', 3:N, sqrt_covm_errf(1, 3:200),'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Etimation Error', 'Filtration Error Covariance Matrix', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Errors', 'FontSize', 30)

%% Point 14

sigma2_a = 1;
[x, z] = data_gen(N, T, sigma2_n, sigma2_a);
[Z_filt, ~, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);

figure(141)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

figure(142)
plot(1:N , K_arr(1,:),'g', 'LineWidth', 1.2)
grid on; grid minor
legend('Gain', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Gain', 'FontSize', 30)
grid on

sigma2_a = 0.2^2;
[x, z] = data_gen(N, T, sigma2_n, sigma2_a);
[Z_filt, X_pred, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);

figure(143)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

figure(144)
plot(1:N , K_arr(1,:), 'g', 'LineWidth', 1.2)
grid on; grid minor
```

```
legend('Gain', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Gain', 'FontSize', 30)


%% Point 15
X0 = [100; 5];

for i = 1:M
    [x, z] = data_gen(N, T, sigma2_n, sigma2_a); %sigma2_a = 0.2^2;
    [Z_filt, X_pred, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0); %sigma2_a = 0;
    err_filt(i,1,:) = (x - Z_filt).^2;
end
Final_ErrFiltered_1 = sqrt( 1/(M-1) * sum(err_filt) );

figure(151)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'b', 'LineWidth', 1.2)
grid on; grid minor
legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

K_underestimated = K_arr(1,N)/5;
for i = 1:M
    [x, z] = data_gen(N, T, sigma2_n, sigma2_a); %sigma2_a = 0.2^2;
    [Z_filt, X_pred, ~ ] = KF_und(z, m, sigma2_a, X0, P0, K_underestimated); %sigma2_a = 0;
    err_filt(i,1,:) = (x - Z_filt).^2;
end
Final_ErrFiltered_2 = sqrt( 1/(M-1) * sum(err_filt) );

figure(152)
plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
grid on; grid minor
legend('True Data','Measurements','Filtered Underestimated Gain','Location','best', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)

figure(153);
hold on
plot(3:N, Final_ErrFiltered_1(1,3:N),'m', 'LineWidth', 1.2)
plot(3:N, Final_ErrFiltered_2(1,3:N),'k', 'LineWidth', 1.2)
grid on; grid minor
legend('Filtered Estimate Error (Optimal)',...
    'Filtered Estimate Error (Underestimated)', 'FontSize', 30)
xlabel('Step', 'FontSize', 30)
ylabel('Error', 'FontSize', 30)
```

```matlab


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%                                FUNCTION                                 %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% data_gen generates trajectory and measurement
% Kalman_filter and Kalman_filter_2 compute kalman filter algorithm with
% different inputs and outputs
% KF_und computes the kalman filter algorithm, with given K (K_underestimated)

function [X, Z] = data_gen(N,T,sigma2_n,sigma2_a)

X = zeros(1,N); % true data
V = zeros(1,N); % velocity
Z = zeros(1,N); % measurments

n = randn*sqrt(sigma2_n); %random noise of measurments

% Initial data
X(1) = 5;
V(1) = 1;
Z(1) = X(1) + n; % first measurment

for i = 2:N
    a = randn*sqrt(sigma2_a); % normally distributed random acceleration
    n = randn*sqrt(sigma2_n); % random noise of measurments
    V(i) = V(i-1) + a*T;
    X(i) = X(i-1) + V(i-1)*T + a*T^2/2;
    Z(i) = X(i) + n;
end

end

function X_Kalman = Kalman_filter(Z,T,m,sigma2_n,sigma2_a)

N = length(Z);
Fi = [1 T; 0 1]; %transition matrix
G = [T/2;T];     %input matrix
H = [1 0];       %observation matrix

% Kalman filter
% Initialization
X = zeros(2,2,N+1); %true data
P = zeros(2,4,N+1); %filtration error covariance matrix
K = zeros(2,N);
X_f = zeros(2,N);

```

```
% Initial conditions
X(:,:,1) = [2,0;0,0];
P(:,1:2,1) = [10000 0; 0 10000];
Q = G*G.'*sigma2_a;
R = sigma2_n;

for i = 2:N+1
    % Prediction of state vector at time i using i-1 measurements
    X(:,2,i-1) = Fi*X(:,1,i-1);
    X_f(:,i-1) = Fi^m*X(:,1,i-1);
    % Prediction error covariance matrix
    P(:,3:4,i-1) = Fi*P(:,1:2,i-1)*Fi.'+Q;
    % Filter gain, weight of residual
    K(:,i-1) = P(:,3:4,i-1)*H.'*(H*P(:,3:4,i-1)*H.'+R)^(-1);
    % Improved estimate by incorporating a new measurement
    X(:,1,i) = X(:,2,i-1)+K(:,i-1)*(Z(i-1)-H*X(:,2,i-1));
    % Filtration error covariance matrix
    P(:,1:2,i) = (eye(2)-K(:,i-1)*H)*P(:,3:4,i-1);
end

X_Kalman(1,1:N) = X(1,1,2:N+1); % filtered data
X_Kalman(2,1:N) = X(1,2,1:N); % predictions
X_Kalman(3,1:N-6) = X_f(1,1:N-6); % predictions 7 steps ahead
X_Kalman(4,1:N) = sqrt(P(1,1,2:N+1)); % error of filtration
X_Kalman(5,1:N) = sqrt(P(1,3,1:N)); % error of prediction
X_Kalman(6,1:N) = K(1,:); % filter gain

end


function [Z_f, X_forecast, P, K] = Kalman_Filter_2(z, m, sigma2_a, X0, P0)
    size_ = length(z);
    T = 1; sigma2_n=20^2;
    X = zeros(2, 1, size_ + 1);  %State vectors of real data
    P = zeros(2, 2, size_ + 1);  %Initial filtration error covariance matrix
    Fi = [1 T; 0 1];
    G = [0.5*T^2; T];
    H = [1 0]; %H

    X(:,:, 1) = X0; %Initian state vector
    P(:, :, 1) = P0;
    Q = sigma2_a*(G*G'); %Covariance matrix of state noise
    R = sigma2_n; %Covariance matrix of measurements noise

    Z_f = zeros(1, size_);
    X_forecast = zeros(1, size_);
    K = zeros(2, 1, size_);
    for i = 2:size_ + 1
        %Prediction part
        X_pred = Fi*X(:,:,i-1);
```

```
        temp =(Fi^(m-1))*X(:,:,i-1);
        X_forecast(1,i-1) = temp(1, :);
        P_pred = ...
            Fi*P(:,:,i - 1)*Fi' + Q;
        %Filtrarion part
        K(:,:,i-1) = P_pred*(H') * ...
            (H*P_pred*H' + R)^(-1);
        X(:,:,i) = X_pred + K(:,:,i-1)*(z(i-1) - H*X_pred);
        Z_f(i-1) = X(1,1,i);
        P(:, :, i) = (eye(2)-K(:,:,i-1)*H)*P_pred;

%         Use for the 15th point, where K_understimated is given
%         %Filtrarion part
%         X(:,:,i) = X_pred + K*(z(i-1) - H*X_pred);
%         Z_f(i-1) = X(1,1,i);
%         P(:, :, i) = (eye(2)-K*H)*P_pred;
    end
end

function [Z_f, X_forecast, P, K] = KF_und(z, m, sigma2_a, X0, P0, K)

N = length(z);
    T = 1;
    X = zeros(2, 1, N + 1);  % State vectors of real data
    P = zeros(2, 2, N + 1);  % Initial filtration error covariance matrix
    Fi = [1 T; 0 1];
    G = [0.5*T^2; T];
    H = [1 0];

    X(:,:, 1) = X0; % Initian state vector
    P(:, :, 1) = P0;
    Q = sigma2_a*(G*G'); % Covariance matrix of state noise
    Z_f = zeros(1, N);
    X_forecast = zeros(1, N);
    for i = 2:N + 1
        %Prediction part
        X_pred = Fi*X(:,:,i-1);
        temp =(Fi^(m-1))*X(:,:,i-1);
        X_forecast(1,i-1) = temp(1, :);
        P_pred = Fi*P(:,:,i - 1)*Fi' + Q;
        %Filtrarion part
        X(:,:,i) = X_pred + K*(z(i-1) - H*X_pred);
        Z_f(i-1) = X(1,1,i);
        P(:, :, i) = (eye(2)-K*H)*P_pred;
    end
end
```