```matlab
%% Development of optimal smoothing to increase the estimation accuracy

% 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');

%% Initial data

N = 200; % observation interval
sigma2_n = 20^2; % variance of noise
sigma2_a = 0.2^2; % variance of acceleraration
T=1; % Period of step

M = 500; %number of runs
X0 = [2; 0];
P0 = [10000 0; 0 10000];
SqrError_X = zeros(M, 1, N - 2); % Squared error of coordinate
SqrError_V = zeros(M, 1, N - 2); % Squared error of velocity

for i=1:M
    [X, z, V] = data_gen(N,T,sigma2_n,sigma2_a);
    % Kalman filter
    [Z_f, X_f, P_pred, P] = kalman(z,  sigma2_a, X0, P0);
    % Smoothing filtered data
    [Z_smoothed, P_smooth] = kalman_sm(Z_f, P_pred, P, T);
    SqrError_X(i,:) = ( X(3:N) - Z_smoothed(1,4:N+1) ).^2;
    SqrError_V(i,:) = ( V(3:N) - Z_smoothed(2,4:N+1) ).^2;
    SqrError_X_NOTsm(i,:) = ( X(1:N) - Z_f(1,2:N + 1) ).^2;
    SqrError_V_NOTsm(i,:) = ( V(1:N) - Z_f(2,2:N + 1) ).^2;
end

Final_ErrSmoothed_X = sqrt( 1/(M-1) * sum(SqrError_X) ); %true estimation error of
coordinate smoothed
Final_ErrSmoothed_V = sqrt( 1/(M-1) * sum(SqrError_V) ); %true estimation error of
velocity smoothed

Final_ErrNOTSmoothed_X = sqrt( 1/(M-1) * sum(SqrError_X_NOTsm) ); %true estimation
error of coordinate
Final_ErrNOTSmoothed_V = sqrt( 1/(M-1) * sum(SqrError_V_NOTsm) ); %true estimation
error of velocity

SmoothError_X = sqrt(P_smooth(1,1,4:N + 1)); %smoothed error of coordinate
SmoothError_V = sqrt(P_smooth(2,2,4:N + 1)); %smoothed error of velocity

FilteredError_X = sqrt(P(1,1,4:N + 1)); %filtered error of coordinate
FilteredError_V = sqrt(P(2,2,4:N + 1)); %filtered error of velocity
```

```
t = 1:N;
figure(1)
plot(t, X, 'g', t, z, 'm-', t, Z_f(1,2:N + 1), 'k', t, Z_smoothed(1,2:N + 1), 'c',↙
'LineWidth', 1.2)
grid on; grid minor
legend('True Data', 'Measurements', 'Filtered Data', 'Smoothed Data', 'FontSize',↙
20);
xlabel('Step', 'FontSize', 30)
ylabel('Data', 'FontSize', 30);

figure(2)
plot(3:N, Final_ErrSmoothed_X(1,:), 1:N, Final_ErrNOTSmoothed_X(1,:), ...
3:N, SmoothError_X(1,:), 3:N, FilteredError_X(1,:), 'LineWidth', 1.2)
grid on; grid minor
legend('True Smoothed Estimation Error', 'True Estimation Error', ...
    'Smoothing Algorithm Error', 'Filtration Error','FontSize', 20);
xlabel('Step', 'FontSize', 30);
ylabel('Error', 'FontSize', 30)
ylim([0 15])


figure(3)
plot(3:N, Final_ErrSmoothed_V(1,:), 1:N, Final_ErrNOTSmoothed_V(1,:), ...
3:N, SmoothError_V(1,:), 3:N, FilteredError_V(1,:), 'LineWidth', 1.2)
grid on; grid minor
legend('True Smoothed Estimation Error', 'True Estimation Error', ...
    'Smoothing Algorithm Error', 'Filtration Error','FontSize', 20);
xlabel('Step', 'FontSize', 30);
ylabel('Error', 'FontSize', 30);
ylim([0 2])


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

function [X, Z, V] = 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 [Z_f, X_f, P_pred, P, K] = kalman(z, sigma2_a, X0, P0)

N = length(z);
T = 1; sigma2_n=20^2;
Z_f = zeros(2, N + 1);                    % State vectors of real data
P = zeros(2, 2, N + 1);                   % Filtration error covariance matrix
P_pred = zeros(2, 2, N + 1);              % Prediction error covariance matrix
Fi = [1 T; 0 1];
G = [0.5*T^2; T];
H = [1 0];

Z_f(:, 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

X_f = zeros(1, N);
K = zeros(2, 1, N);

for i = 2:N + 1
    %Prediction part
    X_pred = Fi*Z_f(:,i-1);
    P_pred(:,:,i-1) = Fi * P(:,:,i - 1) * Fi' + Q;
    %Filtrarion part
    K(:,:,i-1) = P_pred(:,:,i-1)*(H') * (H*P_pred(:,:,i-1)*H' + R)^(-1);
    Z_f(:,i) = X_pred + K(:,:,i-1)*(z(i-1) - H*X_pred);
    P(:, :, i) = (eye(2)-K(:,:,i-1)*H)*P_pred(:,:,i-1);
end

end

function [Z_sm, P_sm] = kalman_sm(data, P_pred, P, T)

N = length(data);
Fi = [1 T; 0 1];
A = zeros(2,2,N - 1);  %coefficient
P_sm = zeros(2, 2, N); %smoothing error covariance matrix
Z_sm = zeros(2,N);     %smoothed data

P_sm(:,:,N) = P(:,:,N);
Z_sm(:,N) = data(:,N);

for i = N - 1:-1:1
    A(:,:,i) = P(:,:,i)*transpose(Fi)*inv(P_pred(:,:,i));
    P_sm(:,:,i) = P(:,:,i) + A(:,:,i)*(P_sm(:,:,i+1) - P_pred(:,:,i))*A(:,:,i).';
```

```
    Z_sm(:,i) = data(:,i) + A(:,:,i)*(Z_sm(:,i+1) - Fi*data(:,i));
end

end
```