```
%% Joint Assimilation of Navigation Datan Coming from Different Sources

% 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');

%%

T=2;
N=500;

InititalState = [1000;100;1000;100]; %[x0; Vx0; y0; Vy0]

sigma_a = 0.3;          % variance of acceleration noise
sigma_D = 50;           % variance of range noise of measurements
sigma_b = 0.004;        % variance of azimuth noise of measurements
sigma_b_add = 0.001;    %

% Creating of arrays for running Kalman filter M times

M=500;                              %number of runs
err_range_filt  = zeros(M,N); %Filtraion error of range
err_range_pred  = zeros(M,N); %Predicted error of range
err_azimut_filt = zeros(M,N); %Filtraion error of azimut
err_azimut_pred = zeros(M,N); %Predicted error of azimut

for i=1:M
    %Generation of deterministic trajectory and its measurements
    [polar,cart,z_p] = ...
        true_traj(N,T,sigma_D,sigma_b,InititalState, sigma_a, sigma_b_add);

    %Applying or Kalman filter for measurements
    [Z_filtered,Z_forecast, range_fe, azimuth_fe] = Kalman_extended(z_p,...
        T,sigma_D,sigma_b, sigma_a, sigma_b_add);

    err_range_filt(i,:) = (polar(1,:) - range_fe(1,:)).^2;
    err_range_pred(i,:) = (polar(1,:) - range_fe(2,:)).^2;
    err_azimut_filt(i,:)= (polar(2,:) - azimuth_fe(1,:)).^2;
    err_azimut_pred(i,:)= (polar(2,:) - azimuth_fe(2,:)).^2;
end

err_range_filt =sqrt(1/(M-1)*sum(err_range_filt));
err_range_pred =sqrt(1/(M-1)*sum(err_range_pred));
err_azimut_filt = sqrt(1/(M-1)*sum(err_azimut_filt));
err_azimut_pred = sqrt(1/(M-1)*sum(err_azimut_pred));

%% Point 6
```

```
 54
 55 % True trajectory, measurements and filtered&extrapolated
 56 figure(1)
 57 polarplot(polar(2,:),polar(1,:), 'm',...
 58     z_p(2,:),z_p(1,:),'c.', ...
 59     azimuth_fe(1,:), range_fe(1,:), 'k')
 60 legend('True trajectory', 'Measurements', 'Filtered', 'FontSize', 30)
 61 grid on; grid minor
 62
 63 %% Point 7
 64
 65 figure(2)
 66 plot(4:N,err_range_filt(4:N),'m',...
 67      4:N,err_range_pred(4:N),'c');
 68 legend('Filtration error','Extrapolation error');
 69 %title('Errors of range');
 70 xlabel('Step')
 71 ylabel('Errors')
 72 grid on; grid minor
 73
 74 %% Point 7
 75
 76 figure(3)
 77 plot(4:N,err_azimut_filt(4:N),'m', ...
 78      4:N,err_azimut_pred(4:N),'c');
 79 legend('Filtration error','Extrapolation error');
 80 %title('Errors of azimiuth');
 81 xlabel('Step')
 82 ylabel('Errors')
 83 grid on; grid minor
 84
 85 %% Point 7
 86
 87 figure(4)
 88 plot(cart(1,:),cart(3,:), 'm',...
 89     Z_filtered(1,:),Z_filtered(3,:),'c')
 90 %title(True and filtered trajectory')
 91 legend('True', 'Filtered')
 92 grid on; grid minor
 93 xlabel('Step')
 94 ylabel('Data')
 95
 96 %% Point 8
 97 % A = NaN;
 98 % b = z_p(1,:);
 99 % b(isnan(A))=0;
100
101 z_pp = z_p(1,1:2:N);
102
103 figure(7)
104 plot(1:N, polar(1,:),'m', 1:2:N, z_pp, 'c',1:N ,range_fe(1,:), 'k')
105 %title('Range')
106 legend('True', 'Measurements', 'Filtered', 'location', 'best','FontSize', 25)
```

```
107 grid on; grid minor
108 xlabel('Step')
109 ylabel('Data')
110
111 %% Point 8
112
113 figure(8)
114 plot(1:N, polar(2,1:N),'m', 4:N, z_p(2,4:N), 'c', 1:N, azimuth_fe(1,:), 'k')
115 %title('Azimuth')
116 legend('True', 'Measurements', 'Filtered')
117 grid on; grid minor
118 xlabel('Step')
119 ylabel('Data')
120
121
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
123 %                                                                                 %
124 %                                   FUNCTION                                      %
125 %                                                                                 %
126 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127
128 function [polar,cartesian, z_polar] = true_traj(N,T,sigma_D,sigma_b,↙
InititalState, sigma_a, sigma_beta_add)
129 % Generation of true trajectory
130
131     X   = zeros(1,N); X(1)   = InititalState(1);
132     Y   = zeros(1,N); Y(1)   = InititalState(3);
133     V_x = zeros(1,N); V_x(1) = InititalState(2);
134     V_y = zeros(1,N); V_y(1) = InititalState(4);
135     a_x = zeros(1,N - 1);
136     a_y =zeros(1,N - 1);
137
138     for i=2:N
139         a_x(i-1) = randn * sigma_a;
140         a_y(i-1) = randn * sigma_a;
141         V_x(i)=V_x(i-1) + a_x(i-1)*T;
142         V_y(i)=V_y(i-1) + a_y(i-1)*T;
143         X(i)=X(i-1)+V_x(i-1)*T + 0.5*a_x(i-1)*T^2;
144         Y(i)=Y(i-1)+V_y(i-1)*T + 0.5*a_y(i-1)*T^2;
145     end
146     cartesian = [X; V_x; Y; V_y];
147
148     %Generation of true values of range D and aimuth b
149     D = sqrt(X.^2+Y.^2);
150     b = atan(X./Y);
151     polar = [D;b];
152
153     %Generation of measurements
154     D_m=zeros(1,N); %array of measurements of range
155     b_m=zeros(1,N); %array of measurements of azimuth
156
157     for i=1:2:N-1
158         D_m(i)=D(i)+randn*sigma_D;
```

```
159         b_m(i)=b(i)+randn*sigma_b;
160     end
161
162     for i=4:2:N
163         D_m(i)=NaN;
164         b_m(i)=b(i)+randn*sigma_beta_add;
165     end
166
167     z_polar = [D_m; b_m];
168
169 end
170
171 function [Z_filtered, Z_forecast, range_fe, azimuth_fe] ...
172     = Kalman_extended(Z_polar, T, sigma_D,sigma_b, sigma_a, sigma_b_add)
173
174 %   Description:
175     size_ = length(Z_polar);
176     Z_filtered = zeros(4, size_);           %Filtered data
177     Z_forecast = zeros(4, size_);           %Forecast data [x; Vx; y; Vy]
178     FiltrErr_CovMatr = zeros(4, 4, size_);  %Filtration error covariance matrix
179     PredErr_CovMatr  = zeros(4, 4, size_); %Prediction error covariance matrix
180     K = zeros(4, 2, size_);
181
182     Z_filtered(:, 3) = ...
183         [Z_polar(1,3)*sin(Z_polar(2,3));...
184         (Z_polar(1,3)*sin(Z_polar(2,3)) - Z_polar(1,1)*sin(Z_polar(2,1)))/↙
(2*T);...
185         Z_polar(1,3)*cos(Z_polar(2,3));...
186         (Z_polar(1,3)*cos(Z_polar(2,3)) - Z_polar(1,1)*cos(Z_polar(2,1)))/↙
(2*T)]; %Initian state vector
187     FiltrErr_CovMatr(:, :, 3) = [10^4 0 0 0; 0 10^4 0 0;...
188         0 0 10^4 0; 0 0 0 10^4];
189
190     TransMatr = [1 T 0 0; 0 1 0 0; 0 0 1 T; 0 0 0 1]; %Ô
191     InputMatr = [0.5*T^2 0; T 0; 0 0.5*T^2; 0 T]; %G
192     CovMatr_StateNoise   = (InputMatr*InputMatr')*sigma_a^2; %Covariance matrix↙
of state noise
193
194     range_fe = zeros(2,size_); %array of filtered and extrapolated range
195     azimuth_fe = zeros(2,size_); %array of filtered and extrapolated azimuth
196     for i = 4:1:size_
197         %Forecasting
198         Z_forecast(:,i-1) =  TransMatr*Z_filtered(:,i-1);
199         PredErr_CovMatr(:,:,i-1) = ...
200             TransMatr*FiltrErr_CovMatr(:,:,i-1)*TransMatr' +↙
CovMatr_StateNoise;
201         h = [sqrt(Z_forecast(1,i-1)^2 + Z_forecast(3,i-1)^2 );...
202              atan(Z_forecast(1,i-1)/Z_forecast(3,i-1)) ];
203         dhdx = zeros(2, 4);
204         dhdx(1,1) =  Z_forecast(1,i-1)/sqrt( (Z_forecast(1,i-1))^2 +↙
(Z_forecast(3,i-1))^2 );
205         dhdx(1,3) =  Z_forecast(3,i-1)/sqrt( (Z_forecast(1,i-1))^2 +↙
(Z_forecast(3,i-1))^2 );
```

```
206         dhdx(2,1) =  Z_forecast(3,i-1)/(Z_forecast(1,i-1)^2 + Z_forecast(3,i-1)^2 );
207         dhdx(2,3) = -Z_forecast(1,i-1)/(Z_forecast(1,i-1)^2 + Z_forecast(3,i-1)^2 );
208
209         if mod(i,2) == 0
210             CovMatr_MeasureNoise = [sigma_D^2 0; 0 sigma_b_add^2];
211         else
212             CovMatr_MeasureNoise = [sigma_D^2 0; 0 sigma_b^2];
213         end
214
215         %Filtrarion part
216         K(:,:,i) = PredErr_CovMatr(:,:,i-1)*(dhdx') * ...
217             (dhdx*PredErr_CovMatr(:,:,i-1)*dhdx' + CovMatr_MeasureNoise)^(-1);
218
219         FiltrErr_CovMatr(:, :, i) = (eye(4)-K(:,:,i)*dhdx)*PredErr_CovMatr(:,:,i-1);
220         Z_temp = Z_polar(:,i);
221         if isnan(Z_temp(1))
222             Z_temp(1) = h(1);
223         end
224         Z_filtered(:,i) = Z_forecast(:,i - 1) + ...
225             K(:,:,i)*(Z_temp - h);
226
227         range_fe(1,i)=sqrt( Z_filtered(1,i)^2+Z_filtered(3,i)^2 );
228         range_fe(2,i)=sqrt( Z_forecast(1,i-1)^2+Z_forecast(3,i-1)^2);
229         azimuth_fe(1,i)=atan(Z_filtered(1,i)/Z_filtered(3,i));
230         azimuth_fe(2,i)=atan(Z_forecast(1,i-1)/Z_forecast(3,i-1));
231     end
232     azimuth_fe(1,1:3) = azimuth_fe(1,4);
233 end
234
235
```