

```
1 %% Tracking of a moving object which trajectory is disturbed by random acceleration
2
3 % Written by Irina Yareshko and Luca Breggion, Skoltech 2022
4
5 close all
6 clear
7 clc
8
9 set(0,'defaulttextInterpreter','latex');
10 set(groot,'defaultAxesTickLabelInterpreter','latex');
11 set(groot,'defaultLegendInterpreter','latex');
12
13 %% 1-2) Generate a true trajectory  $X_i$  of an object motion disturbed by normally
distributed random ...
14 %         acceleration
15
16 %Creating of arrays
17 x = zeros(1, 200); %true data
18 V = zeros(1, 200); %velocity
19 Z = zeros(1, 200); %measurments
20
21 sigma2_n = 20^2;
22 n = randn*sqrt(sigma2_n); %random noise of measurments
23 %Initial data
24 N = 200;
25 T = 1;
26 x(1) = 5;
27 V(1) = 1;
28 Z(1) = x(1) + n; %the first measurment
29
30 %Generation of data
31 for i=2:N
32     sigma2_a = 0.2^2;
33     a = randn*sqrt(sigma2_a); %normally distributed random acceleration
34     n = randn*sqrt(sigma2_n); %random noise of measurments
35     V(i) = V(i - 1) + a*T;
36     x(i) = x(i - 1) + V(i - 1)*T + a*T^2/2;
37     Z(i) = x(i) + n;
38 end
39
40 %% 3-4) Presenting the system at state space and state vector estimations
41
42 %State matrixes
43 Fi = [1 T; 0 1]; %transition matrix
44 G = [T/2; T]; %input matrix
45 H = [1 0]; %observation matrix
46
47 % Kalman filter
48 X = zeros(2, 2, N + 1); %true data
49 P = zeros(2, 4, N + 1); %filtration error covariance matrix
```

```

50 K = zeros(2, N);
51 X_f = zeros(2, N);
52
53 %Initial conditions
54 X(:, :, 1) = [2, 0; 0, 0];
55 P(:, 1:2, 1) = [10000 0; 0 10000];
56 Q = G*G.'*sigma2_a;
57 R = sigma2_n;
58
59 for i = 2:(N + 1)
60     %Prediction of state vector at time i using i-1 measurements
61     X(:, 2, i-1) = Fi*X(:, 1, i-1);
62     X_f(:, i-1) = Fi^7*X(:, 1, i-1);
63
64     %Prediction error covariance matrix
65     P(:, 3:4, i-1) = Fi*P(:, 1:2, i-1)*Fi.'+Q;
66
67     %Filter gain, weight of residual
68     K(:, i-1) = P(:, 3:4, i-1)*H.'*(H*P(:, 3:4, i-1)*H.'+R)^(-1);
69
70     %Improved estimate by incorporating a new measurement
71     X(:, 1, i) = X(:, 2, i-1) + K(:, i-1)*(Z(i-1) - H*X(:, 2, i-1));
72
73     %Filtration error covariance matrix
74     P(:, 1:2, i) = (eye(2) - K(:, i-1)*H)*P(:, 3:4, i-1);
75 end
76
77 x_Kalman(1:N) = X(1, 1, 2:N+1);
78
79 %% 5) Building of plots of true trajectory, measurments, filtered estimates of
80 %     state vector
81
82 t = 1:N; %array of steps
83
84 figure(1)
85 plot(t, x, 'c', t, Z, 'm', t, x_Kalman, 'k', 'LineWidth', 1.2)
86 grid on; grid minor
87 xlabel('Step', 'FontSize', 30)
88 ylabel('Data', 'FontSize', 30)
89 legend('True data', 'Measurments', 'Filtered Estimates of State Vector', 'FontSize', 30);
90
91 %% 6) Filter gain and filtration error
92
93 figure(2)
94 plot(t, K(1, :), 'r', 'LineWidth', 1.2)
95 grid on; grid minor
96 xlabel('Step', 'FontSize', 30)
97 ylabel('Filter Gain K', 'FontSize', 30)
98 legend('Filter Gain', 'FontSize', 30)

```

```

99 ylim([0 1])
100
101 P_sq(1:N)=sqrt(P(1,1,1:N));
102
103 figure(3)
104 plot(t, P_sq,'r', 'LineWidth', 1.2)
105 grid on; grid minor
106 xlabel('Step', 'FontSize', 30)
107 ylabel('Filtration error', 'FontSize', 30)
108 legend('Filtration error', 'FontSize', 30)
109
110 %% 7-8-9 + 10) Estimation of dynamics of mean-squared error of estimation over...
111 %      observation interval
112
113 clear all
114
115 %Initial data
116 m = 7;           % number of steps ahead
117 M = 500;         % number of runs
118 N = 200;         % observation interval
119 sigma2_n = 20^2; % variance of noise
120 sigma2_a = 0.2^2; % variance of acceleration
121 T = 1;          % period of step
122
123 % Initialization
124 Error_X      = zeros(M, N);      % array of errors of filtered estimate
125 Error_X_f    = zeros(M, N);      % array of errors of forecasts
126 Error_X_f7   = zeros(M, N - 6); % array of errors of forecasts
127 X_Kalman     = zeros(6, N, M);   % array of filtered data
128 x           = zeros(2, N);
129
130 for i = 1:M
131     [x, Z] = data_gen(N,T,sigma2_n,sigma2_a);
132     X_Kalman(:,:,i) = Kalman_filter(Z,T,m,sigma2_n,sigma2_a); % Kalman filter
133     Error_X(i,:)    = (x(1,:) - X_Kalman(1,:,i)).^2; % errors of filtered estimate
134     Error_X_f(i,:)  = (x(1,:) - X_Kalman(2,:,i)).^2; % errors of forecasts 1-step
135     Error_X_f7(i,:) = (x(1,7:N) - X_Kalman(3,1:N-6,i)).^2; % errors of forecasts 7-
step
136 end
137
138 %Final average value of Error over M runs
139 Final_Error_X      = sqrt(1/(M-1)*sum(Error_X));
140 Final_Error_X_f    = sqrt(1/(M-1)*sum(Error_X_f));
141 Final_Error_X_f7   = sqrt(1/(M-1)*sum(Error_X_f7));
142
143 %Building of plot of final errors of filtered estimate and errors of
144 %forecasts
145 t=1:(N-2);
146
147 %True error for filtration

```

```
148 figure(4)
149 plot(t,Final_Error_X(3:N),'m',t,X_Kalman(4,3:N,1),'k','LineWidth',1.2);
150 grid on; grid minor
151 xlabel('Step','FontSize',30)
152 ylabel('Errors','FontSize',30)
153 legend('True Estimation Error','Filtration Error Covariance Matrix','FontSize',30);
154
155 t1 = 7:N;
156 figure(6)
157 plot(t, Final_Error_X(3:N),'m', t1, Final_Error_X_f7,'k','LineWidth',1.2);
158 grid on; grid minor
159 xlabel('Step','FontSize',30)
160 ylabel('Errors','FontSize',30)
161 legend('True estimation error','True error of 7-step prediction','FontSize',30)
162
163 t = 1:N;
164 figure(8)
165 plot(t,X_Kalman(6,:,1),'c','LineWidth',1.2);
166 grid on; grid minor
167 xlabel('Step','FontSize',30)
168 ylabel('K','FontSize',30)
169 legend('Filter gain with  $P^{-1}$ ','FontSize',30,'interpreter','latex');
170
171 %% Point 11
172 N = 200;
173 T = 1;
174 X0 = [2; 0];
175 P0 = [10000 0; 0 10000];
176 sigma2_n = 20^2;
177 sigma2_a = 0.2^2;
178 m = 7;
179 M = 500;
180 err_filt = zeros(M, 1, N);
181
182 for i = 1:M
183     [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
184     [Z_filt, ~, P_mat] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
185     err_filt(i,1,:) = (x - Z_filt).^2;
186 end
187
188 %% Pont 12
189 sigma2_a = 0;
190
191 for i = 1:M
192     [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
193     [Z_filt, ~, P_matr, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
194     err_filt(i,1,:) = (x - Z_filt).^2;
195 end
196 fin_err_filt = sqrt( 1/(M-1) * sum(err_filt) );
```

```
197 sqrt_covm_errf = sqrt(P_matr(1,1,2:N + 1));
198
199 figure(121)
200 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
201 grid on; grid minor
202 legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize', 30, 'location', 'best')
203 xlabel('Step', 'FontSize', 30)
204 ylabel('Data', 'FontSize', 30)
205
206 figure(122)
207 plot(1:N, K_arr(1,:), 'g', 'LineWidth', 1.2)
208 grid on; grid minor
209 legend('Filter Gain', 'FontSize', 30)
210 xlabel('Step', 'FontSize', 30)
211 ylabel('Gain K', 'FontSize', 30)
212
213 figure(123)
214 plot(3:N, fin_err_filt(1,3:N), 'm', 3:N, sqrt_covm_errf(1, 3:200), 'k', 'LineWidth', 1.2)
215 grid on; grid minor
216 legend('True Estimation Error','Filtration Error Covariance Matrix', 'FontSize', 30)
217 xlabel('Step', 'FontSize', 30)
218 ylabel('Errors', 'FontSize', 30)
219
220 %% Pont 13
221
222 err_for = zeros(M, 1, N);
223
224 for i = 1:M
225     [x, z] = data_gen(N, T, sigma2_n, 0.2^2); %0.2^2 = sigma2_a;
226     [Z_filt, X_pred, ~, ~] = Kalman_Filter_2(z, m, 0, X0, P0); %Q = 0;
227     err_filt(i,1,:) = (x - Z_filt).^2;
228     err_for(i,1,:) = (x - X_pred).^2;
229 end
230
231 fin_err_filt = sqrt( 1/(M-1) * sum(err_filt) );
232 fin_err_for = sqrt( 1/(M-1) * sum(err_for) );
233
234 figure(131)
235 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'b', 'LineWidth', 1.2)
236 grid on; grid minor
237 legend('True','Measurements','Filtered Estimates of State Vector', 'FontSize', 30)
238 xlabel('Step', 'FontSize', 30)
239 ylabel('Data', 'FontSize', 30)
240
241 figure(132)
242 plot(3:N, fin_err_filt(1,3:N), 'm', 3:N, fin_err_for(1,3:N), 'k', 'LineWidth', 1.2)
243 grid on; grid minor
```

```
244 legend('Filtered Estimate Error', 'Prediction Estimate Error', 'FontSize', 30)
245 xlabel('Step', 'FontSize', 30)
246 ylabel('Errors', 'FontSize', 30)
247
248 figure(133)
249 plot(3:N, fin_err_filt(1,3:N), 'm', 3:N, sqrt_covm_errf(1, 3:200), 'k', 'LineWidth', 1.2)
250 grid on; grid minor
251 legend('True Etimation Error', 'Filtration Error Covariance Matrix', 'FontSize', 30)
252 xlabel('Step', 'FontSize', 30)
253 ylabel('Errors', 'FontSize', 30)
254
255 %% Point 14
256
257 sigma2_a = 1;
258 [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
259 [Z_filt, ~, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
260
261 figure(141)
262 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
263 grid on; grid minor
264 legend('True Data', 'Measurements', 'Filtered Estimates of State Vector', 'FontSize', 30)
265 xlabel('Step', 'FontSize', 30)
266 ylabel('Data', 'FontSize', 30)
267
268 figure(142)
269 plot(1:N, K_arr(1,:), 'g', 'LineWidth', 1.2)
270 grid on; grid minor
271 legend('Gain', 'FontSize', 30)
272 xlabel('Step', 'FontSize', 30)
273 ylabel('Gain', 'FontSize', 30)
274 grid on
275
276 sigma2_a = 0.2^2;
277 [x, z] = data_gen(N, T, sigma2_n, sigma2_a);
278 [Z_filt, X_pred, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0);
279
280 figure(143)
281 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
282 grid on; grid minor
283 legend('True Data', 'Measurements', 'Filtered Estimates of State Vector', 'FontSize', 30)
284 xlabel('Step', 'FontSize', 30)
285 ylabel('Data', 'FontSize', 30)
286
287 figure(144)
288 plot(1:N, K_arr(1,:), 'g', 'LineWidth', 1.2)
289 grid on; grid minor
```

```
290 legend('Gain', 'FontSize', 30)
291 xlabel('Step', 'FontSize', 30)
292 ylabel('Gain', 'FontSize', 30)
293
294
295 %% Point 15
296 X0 = [100; 5];
297
298 for i = 1:M
299     [x, z] = data_gen(N, T, sigma2_n, sigma2_a); %sigma2_a = 0.2^2;
300     [Z_filt, X_pred, ~, K_arr] = Kalman_Filter_2(z, m, sigma2_a, X0, P0); %sigma2_a
= 0;
301     err_filt(i,1,:) = (x - Z_filt).^2;
302 end
303 Final_ErrFiltered_1 = sqrt( 1/(M-1) * sum(err_filt) );
304
305 figure(151)
306 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'b', 'LineWidth', 1.2)
307 grid on; grid minor
308 legend('True Data','Measurements','Filtered Estimates of State Vector', 'FontSize',
30)
309 xlabel('Step', 'FontSize', 30)
310 ylabel('Data', 'FontSize', 30)
311
312 K_underestimated = K_arr(1,N)/5;
313 for i = 1:M
314     [x, z] = data_gen(N, T, sigma2_n, sigma2_a); %sigma2_a = 0.2^2;
315     [Z_filt, X_pred, ~ ] = KF_und(z, m, sigma2_a, X0, P0, K_underestimated); %
sigma2_a = 0;
316     err_filt(i,1,:) = (x - Z_filt).^2;
317 end
318 Final_ErrFiltered_2 = sqrt( 1/(M-1) * sum(err_filt) );
319
320 figure(152)
321 plot(1:N, x, 'g', 1:N, z, 'm-', 1:N, Z_filt, 'k', 'LineWidth', 1.2)
322 grid on; grid minor
323 legend('True Data','Measurements','Filtered Underestimated Gain','Location','best',
'FontSize', 30)
324 xlabel('Step', 'FontSize', 30)
325 ylabel('Data', 'FontSize', 30)
326
327 figure(153);
328 hold on
329 plot(3:N, Final_ErrFiltered_1(1,3:N),'m', 'LineWidth', 1.2)
330 plot(3:N, Final_ErrFiltered_2(1,3:N),'k', 'LineWidth', 1.2)
331 grid on; grid minor
332 legend('Filtered Estimate Error (Optimal)',...
333 'Filtered Estimate Error (Underestimated)', 'FontSize', 30)
334 xlabel('Step', 'FontSize', 30)
335 ylabel('Error', 'FontSize', 30)
```

```
336
337
338 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
339 %                                                                 %
340 %                                FUNCTION                            %
341 %                                                                 %
342 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
343
344 % data_gen generates trajectory and measurement
345 % Kalman_filter and Kalman_filter_2 compute kalman filter algorithm with
346 % different inputs and outputs
347 % KF_und computes the kalman filter algorithm, with given K (K_underestimated)
348
349 function [X, Z] = data_gen(N,T,sigma2_n,sigma2_a)
350
351 X = zeros(1,N); % true data
352 V = zeros(1,N); % velocity
353 Z = zeros(1,N); % measurments
354
355 n = randn*sqrt(sigma2_n); %random noise of measurments
356
357 % Initial data
358 X(1) = 5;
359 V(1) = 1;
360 Z(1) = X(1) + n; % first measurment
361
362 for i = 2:N
363     a = randn*sqrt(sigma2_a); % normally distributed random acceleration
364     n = randn*sqrt(sigma2_n); % random noise of measurments
365     V(i) = V(i-1) + a*T;
366     X(i) = X(i-1) + V(i-1)*T + a*T^2/2;
367     Z(i) = X(i) + n;
368 end
369
370 end
371
372 function X_Kalman = Kalman_filter(Z,T,m,sigma2_n,sigma2_a)
373
374 N = length(Z);
375 Fi = [1 T; 0 1]; %transition matrix
376 G = [T/2;T];      %input matrix
377 H = [1 0];        %observation matrix
378
379 % Kalman filter
380 % Initialization
381 X = zeros(2,2,N+1); %true data
382 P = zeros(2,4,N+1); %filtration error covariance matrix
383 K = zeros(2,N);
384 X_f = zeros(2,N);
385
```



```

386 % Initial conditions
387 X(:, :, 1) = [2, 0; 0, 0];
388 P(:, 1:2, 1) = [10000 0; 0 10000];
389 Q = G*G.'*sigma2_a;
390 R = sigma2_n;
391
392 for i = 2:N+1
393     % Prediction of state vector at time i using i-1 measurements
394     X(:, 2, i-1) = Fi*X(:, 1, i-1);
395     X_f(:, i-1) = Fi^m*X(:, 1, i-1);
396     % Prediction error covariance matrix
397     P(:, 3:4, i-1) = Fi*P(:, 1:2, i-1)*Fi.'+Q;
398     % Filter gain, weight of residual
399     K(:, i-1) = P(:, 3:4, i-1)*H.'*(H*P(:, 3:4, i-1)*H.'+R)^(-1);
400     % Improved estimate by incorporating a new measurement
401     X(:, 1, i) = X(:, 2, i-1)+K(:, i-1)*(Z(i-1)-H*X(:, 2, i-1));
402     % Filtration error covariance matrix
403     P(:, 1:2, i) = (eye(2)-K(:, i-1)*H)*P(:, 3:4, i-1);
404 end
405
406 X_Kalman(1, 1:N) = X(1, 1, 2:N+1); % filtered data
407 X_Kalman(2, 1:N) = X(1, 2, 1:N); % predictions
408 X_Kalman(3, 1:N-6) = X_f(1, 1:N-6); % predictions 7 steps ahead
409 X_Kalman(4, 1:N) = sqrt(P(1, 1, 2:N+1)); % error of filtration
410 X_Kalman(5, 1:N) = sqrt(P(1, 3, 1:N)); % error of prediction
411 X_Kalman(6, 1:N) = K(1, :); % filter gain
412
413 end
414
415
416 function [Z_f, X_forecast, P, K] = Kalman_Filter_2(z, m, sigma2_a, X0, P0)
417     size_ = length(z);
418     T = 1; sigma2_n=20^2;
419     X = zeros(2, 1, size_ + 1); %State vectors of real data
420     P = zeros(2, 2, size_ + 1); %Initial filtration error covariance matrix
421     Fi = [1 T; 0 1];
422     G = [0.5*T^2; T];
423     H = [1 0]; %H
424
425     X(:, :, 1) = X0; %Initian state vector
426     P(:, :, 1) = P0;
427     Q = sigma2_a*(G*G'); %Covariance matrix of state noise
428     R = sigma2_n; %Covariance matrix of measurements noise
429
430     Z_f = zeros(1, size_);
431     X_forecast = zeros(1, size_);
432     K = zeros(2, 1, size_);
433     for i = 2:size_ + 1
434         %Prediction part
435         X_pred = Fi*X(:, :, i-1);

```

```

436     temp = (Fi^(m-1))*X(:, :, i-1);
437     X_forecast(1, i-1) = temp(1, :);
438     P_pred = ...
439         Fi*P(:, :, i-1)*Fi' + Q;
440     %Filtrarion part
441     K(:, :, i-1) = P_pred*(H') * ...
442         (H*P_pred*H' + R)^(-1);
443     X(:, :, i) = X_pred + K(:, :, i-1)*(z(i-1) - H*X_pred);
444     Z_f(i-1) = X(1, 1, i);
445     P(:, :, i) = (eye(2)-K(:, :, i-1)*H)*P_pred;
446
447 %         Use for the 15th point, where K_underestimated is given
448 %         %Filtrarion part
449 %         X(:, :, i) = X_pred + K*(z(i-1) - H*X_pred);
450 %         Z_f(i-1) = X(1, 1, i);
451 %         P(:, :, i) = (eye(2)-K*H)*P_pred;
452     end
453 end
454
455 function [Z_f, X_forecast, P, K] = KF_und(z, m, sigma2_a, X0, P0, K)
456
457 N = length(z);
458     T = 1;
459     X = zeros(2, 1, N + 1); % State vectors of real data
460     P = zeros(2, 2, N + 1); % Initial filtration error covariance matrix
461     Fi = [1 T; 0 1];
462     G = [0.5*T^2; T];
463     H = [1 0];
464
465     X(:, :, 1) = X0; % Initian state vector
466     P(:, :, 1) = P0;
467     Q = sigma2_a*(G*G'); % Covariance matrix of state noise
468     Z_f = zeros(1, N);
469     X_forecast = zeros(1, N);
470     for i = 2:N + 1
471         %Prediction part
472         X_pred = Fi*X(:, :, i-1);
473         temp = (Fi^(m-1))*X(:, :, i-1);
474         X_forecast(1, i-1) = temp(1, :);
475         P_pred = Fi*P(:, :, i-1)*Fi' + Q;
476         %Filtrarion part
477         X(:, :, i) = X_pred + K*(z(i-1) - H*X_pred);
478         Z_f(i-1) = X(1, 1, i);
479         P(:, :, i) = (eye(2)-K*H)*P_pred;
480     end
481 end

```