Table of Contents

Laboratory work 7

Converting a physical distance to a grid distance using least-square

method

Group 5: Ruslan Agishev, Andrei Chemikhin, Valery Nevzorov

Skoltech, 2017

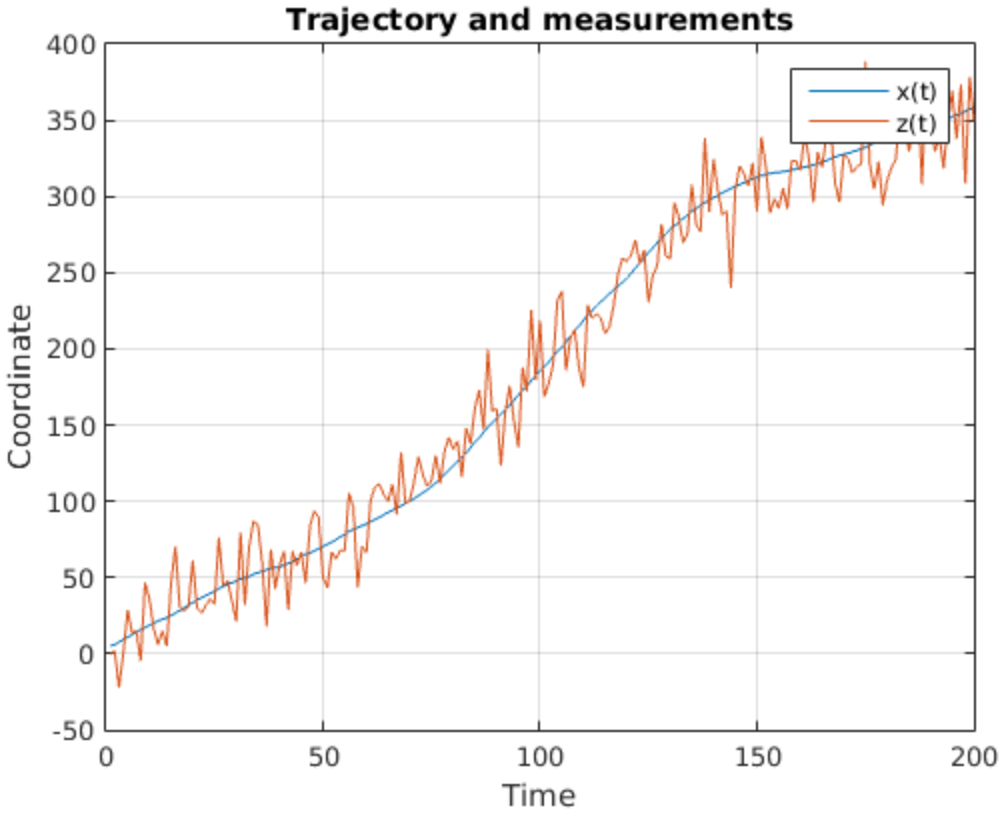
trajectory generation

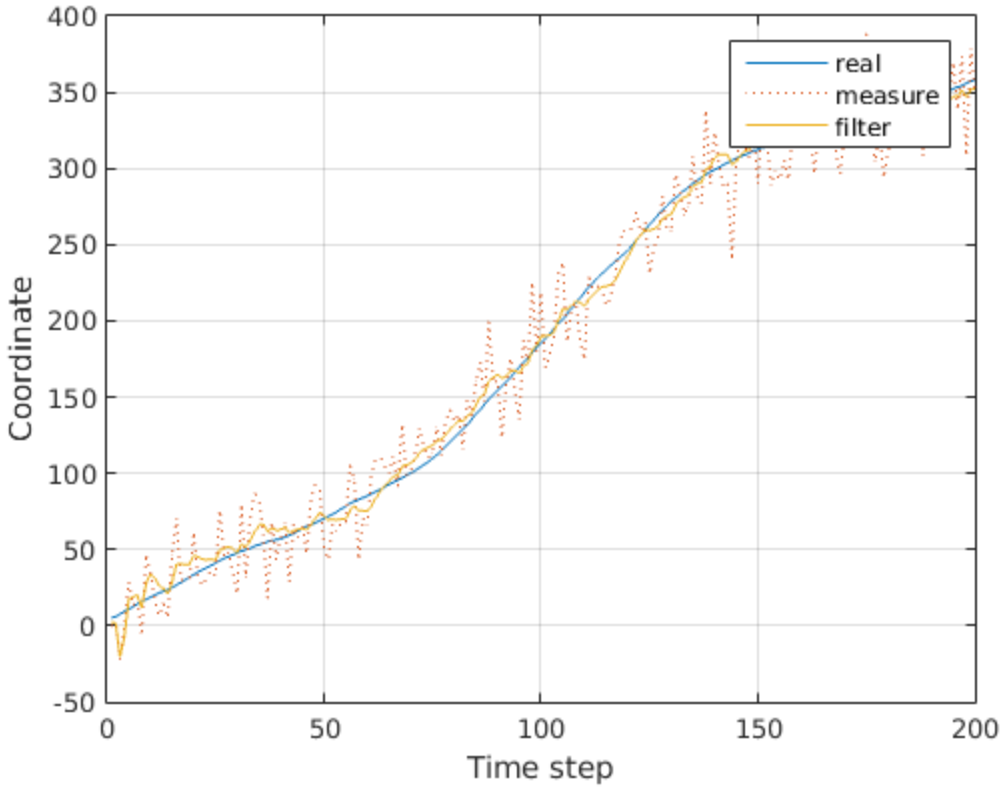
```
close all;
clear;
N = 200;
M=500;
T=1;
v1=1;
sigmaA=0.2;
sigmaN=20;
x1=5;
[x, z, v] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);

t = 1:N;
figure(1)
plot(t,x, t,z)
title('Trajectory and measurements')
legend('x(t)', 'z(t)');
xlabel('Time');
ylabel('Coordinate');
grid on;
```

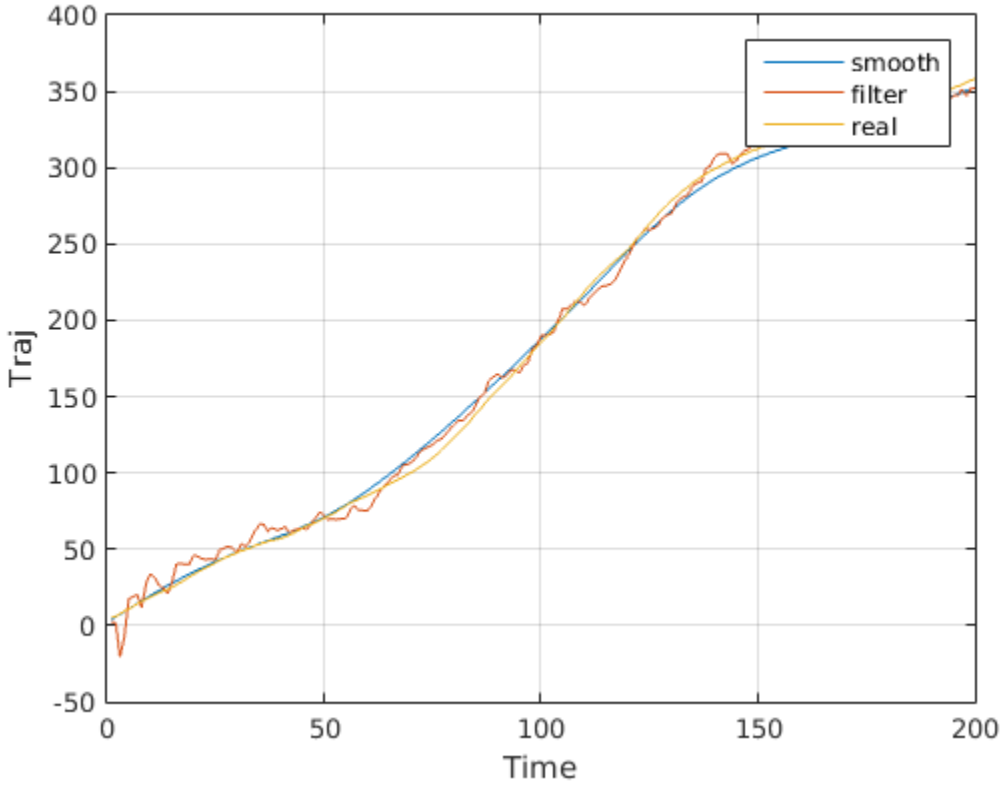
```
% state space - form of equations
[F,G,H] = state_space(T);
% initial covariance matrix
P0 = [10000 0; 0 10000];
X0 = [2;0];
R = sigmaN^2;
Q = G*G'*sigmaA^2;

% filterred values for 1 trajectory
[~,Ppr,Xfl,Pfl1,K] = kalman_filter(X0,P0,F,Q,H,R,z);

figure(2)
plot(t,x, t,z,':', t,Xfl(1,:));
legend('real', 'measure', 'filter');
ylabel('Coordinate')
xlabel('Time step')
grid on;

% backward smoothing algorythm for filtered trajectory
[Xsm, P] = smoothing_back(Xfl, Pfl1, Ppr, F);
xsm = Xsm(1,:);
figure(3)
plot(t,xsm, t,Xfl(1,:), t,x);
legend('smooth', 'filter', 'real');
grid on
xlabel('Time')
ylabel('Traj')

% generation of M=500 random trajectories
X = cell(1,M);
Z = cell(1,M);
V = cell(1,M);
for i=1:M
    [X{i}, Z{i}, V{i}] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
end
% filtrations
Xfl = cell(1,M);
Xpr = cell(1,M);
Xsm = cell(1,M);
Psm = cell(1,M);
Pfl = cell(1,M);
Ppr = cell(1,M);
xsm = cell(1,M);
vsm = cell(1,M);
xfl = cell(1,M);
vfl = cell(1,M);

for i=1:M
    [~,Ppr{i},Xfl{i},Pfl{i},~] = kalman_filter(X0,P0,F,Q,H,R,Z{i});
    xfl{i} = Xfl{i}(1,:);
    vfl{i} = Xfl{i}(2,:);
    [Xsm{i}, Psm{i}] = smoothing_back(Xfl{i}, Pfl{i}, Ppr{i}, F);
    xsm{i} = Xsm{i}(1,:);
```

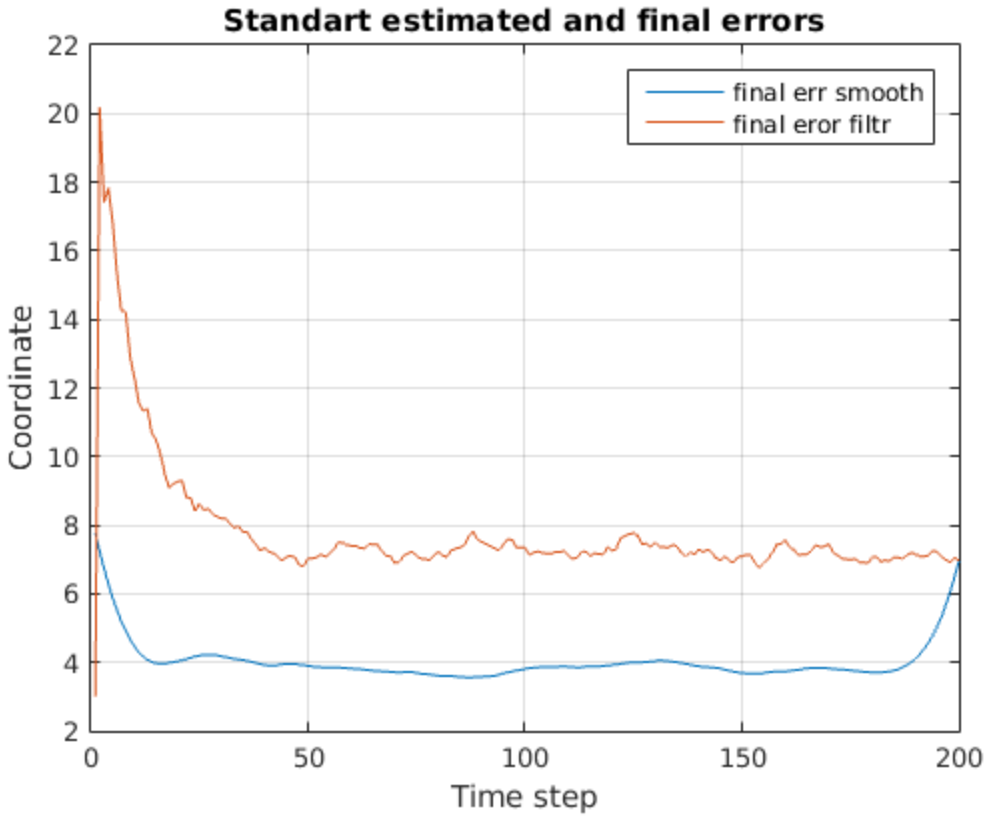
```
    vsm{i} = Xsm{i}(2,:);
end

% estimation of errors between smoothed and real trajectories
fexsm = final_error(xsm, X);
fevsm = final_error(vsm, V);

% estimation of errors between filtered and real trajectories
fex = final_error(xfl, X);
fev = final_error(vfl, V);

% estimation of standart deviation error between filtered and real
% trajectories
px = nan(1,N);
pv = nan(1,N);
for i=1:N
    px(i) = sqrt(Pfl1{i}(1,1));
    pv(i) = sqrt(Pfl1{i}(2,2));
end

% estimation of standart deviation error between smoothed and real
% trajectories
pxsm = nan(1,N);
pvsm = nan(1,N);
for i=1:N
    pxsm(i) = sqrt(P{i}(1,1));
    pvsm(i) = sqrt(P{i}(2,2));
end

figure(4)
plot(t,fexsm, t,fex);
legend('final err smooth', 'final eror filtr');
ylabel('Coordinate')
xlabel('Time step')
title('Standart estimated and final errors')
grid on;

figure(5)
plot(t,pxsm, t,px);
legend('stand.dev', 'filtr error');
ylabel('Coordinate')
xlabel('Time step')
title('Standart estimated and final errors')
grid on;
xlim([4,N])

figure(6)
plot(t,fevsm, t,fev);
legend('final err smooth', 'fin error filter');
ylabel('Velocity')
xlabel('Time step')
title('Standart estimated and final errors')
grid on;
```

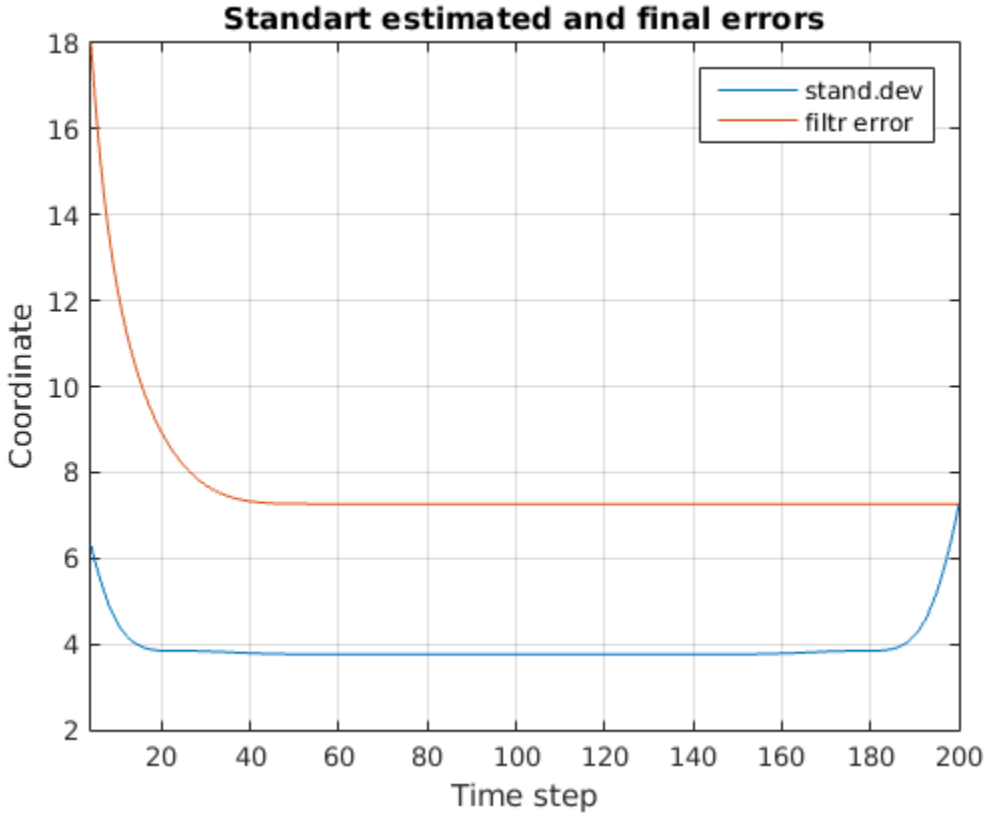
```
figure(7)
plot(t,pvsm, t,pv);
legend('stand.dev vel', 'filtr error');
ylabel('Velocity')
xlabel('Time step')
title('Standart estimated and final errors')
grid on;
xlim([4,N])
```

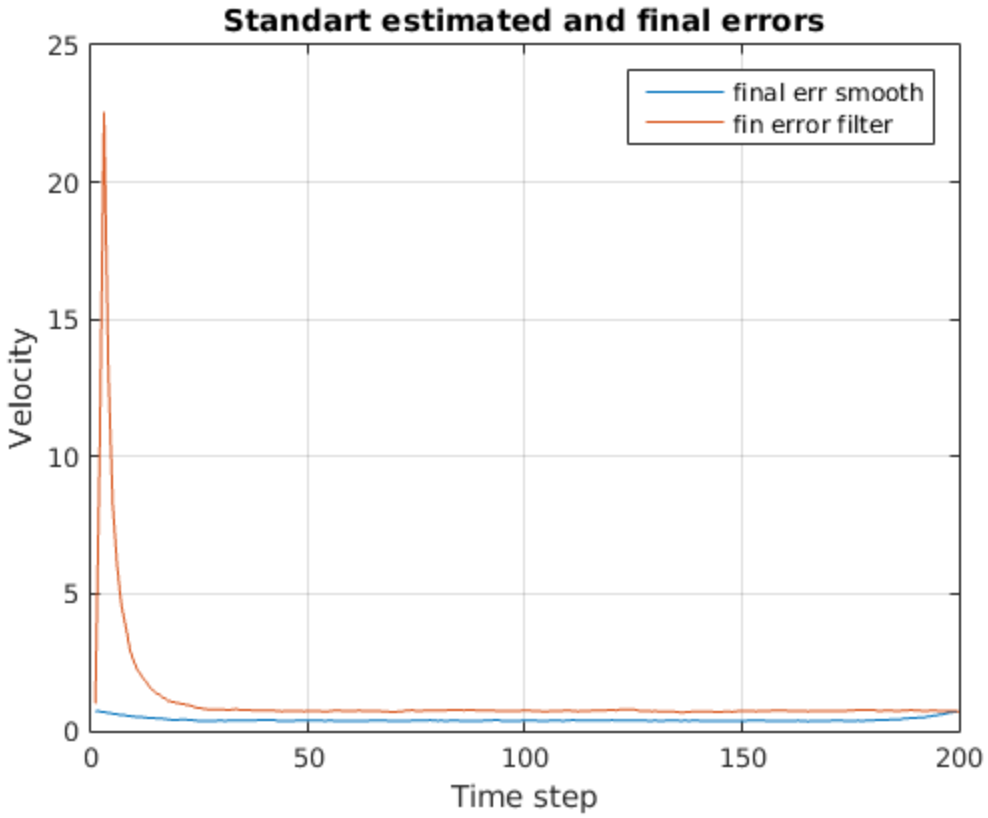


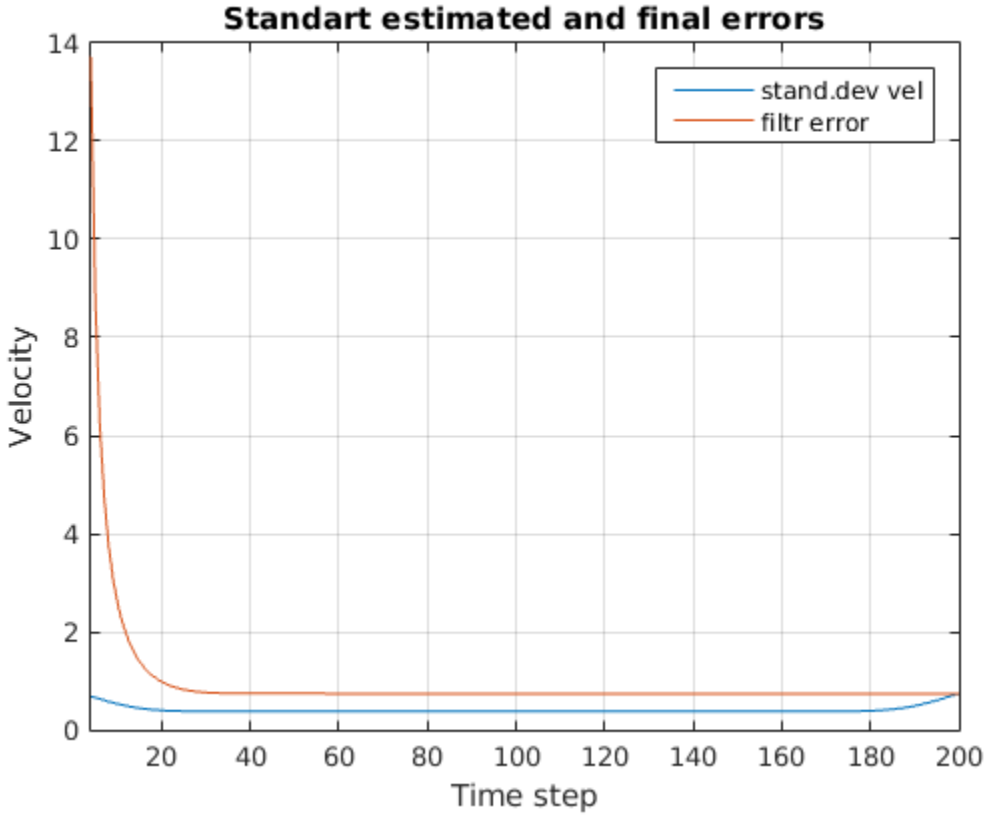

real
measure
filter
Coordinate
Time step
400
350
300
250
200
150
100
50
0
-50
0
50
100
150
200


smooth
filter
real
Traj
Time
400
350
300
250
200
150
100
50
0
-50
0
50
100
150
200


Standart estimated and final errors
final err smooth
final eror filtr
Coordinate
Time step
22
20
18
16
14
12
10
8
6
4
2
0
50
100
150
200


Standart estimated and final errors
stand.dev
filtr error
Coordinate
Time step
18
16
14
12
10
8
6
4
2
20
40
60
80
100
120
140
160
180
200


Standart estimated and final errors
final err smooth
fin error filter
Velocity
Time step
0
5
10
15
20
25
0
50
100
150
200


Standart estimated and final errors
stand.dev vel
filtr error
Velocity
Time step
0
2
4
6
8
10
12
14
20
40
60
80
100
120
140
160
180
200

Conclusion:

Using backward smoothing algorythm we achived better results, smaller error of real trajectory estimation.

Published with MATLAB® R2015a