Laboratory work 13

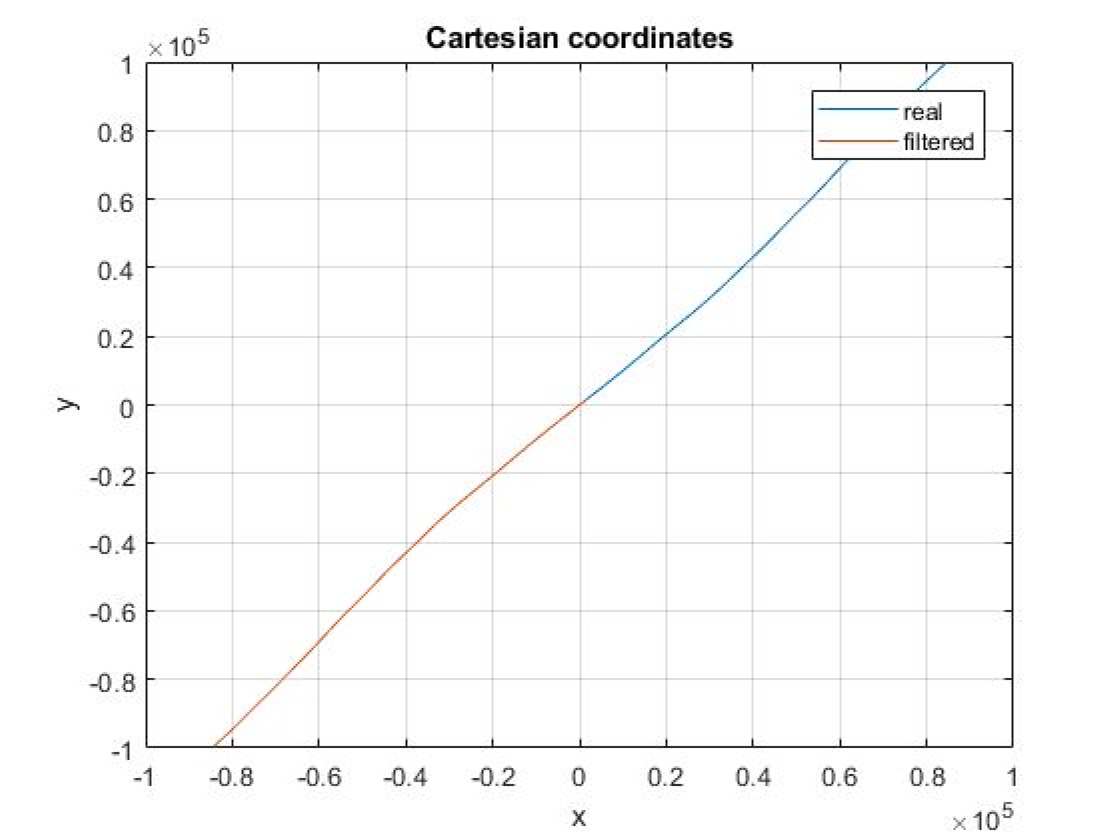
Joint assimilation of navigation data coming from different sources

Group 5: Andrei Chemikhin, Ruslan Agishev, Valery Nevzorov

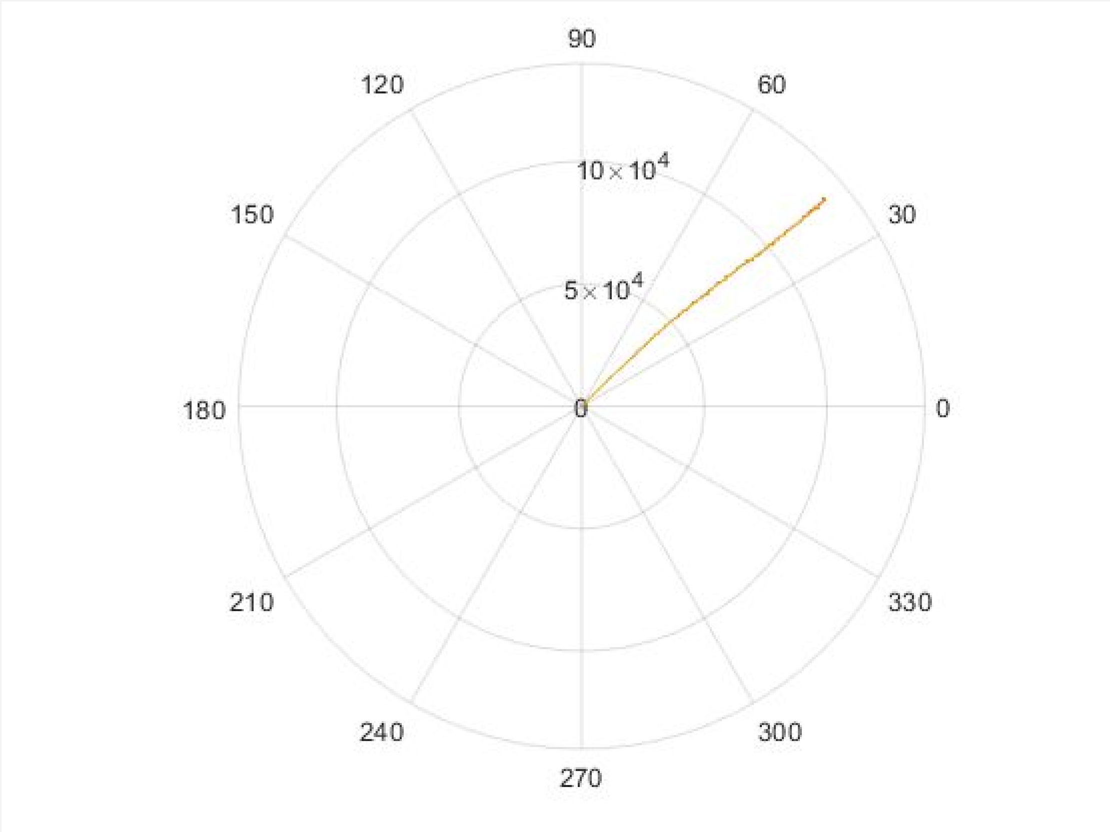
Skoltech, 2017


```
% trajectory generation
close all;
clear;
N = 500;
T = 2;
vx1=100;
vy1=100;
sigmaA = 0.3;
sigmab1 = 0.004;
sigmab2 = 0.001;
sigmaD = 50;
x1 = 1000;
y1 = 1000;
P0 = 10e10*eye(4);

t=1:N;
[x,y, b,D, bm,Dm] = trajgen_acc(sigmaA, N, T, x1,y1,...
    vx1, vy1, sigmab1,sigmab2,sigmaD);

X0 = zeros(4,1);
x1m = Dm(1)*sin(bm(1));
x3m = Dm(3)*sin(bm(3));
y1m = Dm(1)*cos(bm(1));
y3m = Dm(3)*cos(bm(3));
X0(1) = x3m;
X0(2) = (x3m-x1m)/(2*T);
X0(3) = y3m;
X0(4) = (y3m-y1m)/(2*T);


[F,G] = state_space(T);
Q = G*G'*sigmaA^2;
R1 = diag([sigmaD^2 sigmab1^2]);
R2 = sigmab2^2;
```

```
[Xpr,Ppr,Xfl,Pfl,K] = extended_kalman_filter(X0,P0,F,Q,R1,R2,b,D);

xfl = Xfl(1,:);
yfl = Xfl(3,:);

figure
plot(x,y, xfl(1:2:(N-1)),yfl(1:2:(N-1)))
grid on
title('Cartesian coordinates')
xlabel('x')
ylabel('y')
legend('real', 'filtered')
```

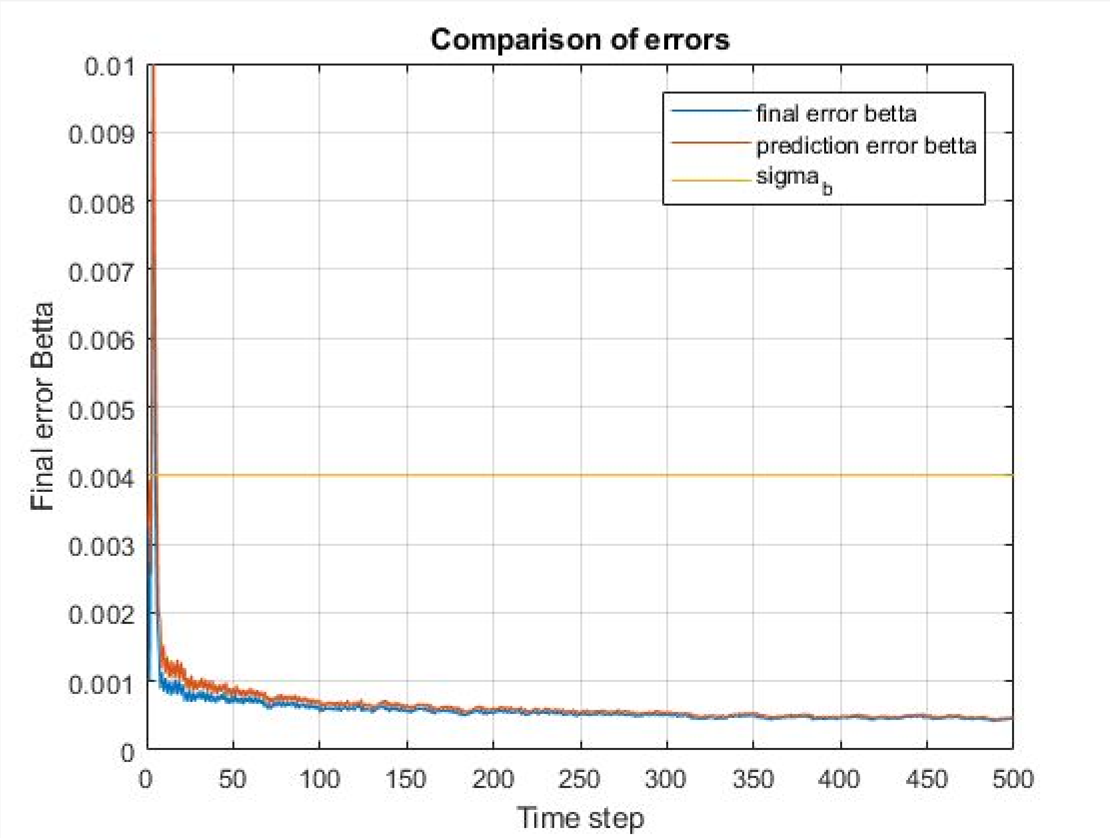



```
[Dfl, bfl] = cartesian2polar(xfl,yfl);

figure
polarplot(b,D, bm(1:2:(N-1)),Dm(1:2:(N-1)), bfl,Dfl)
```

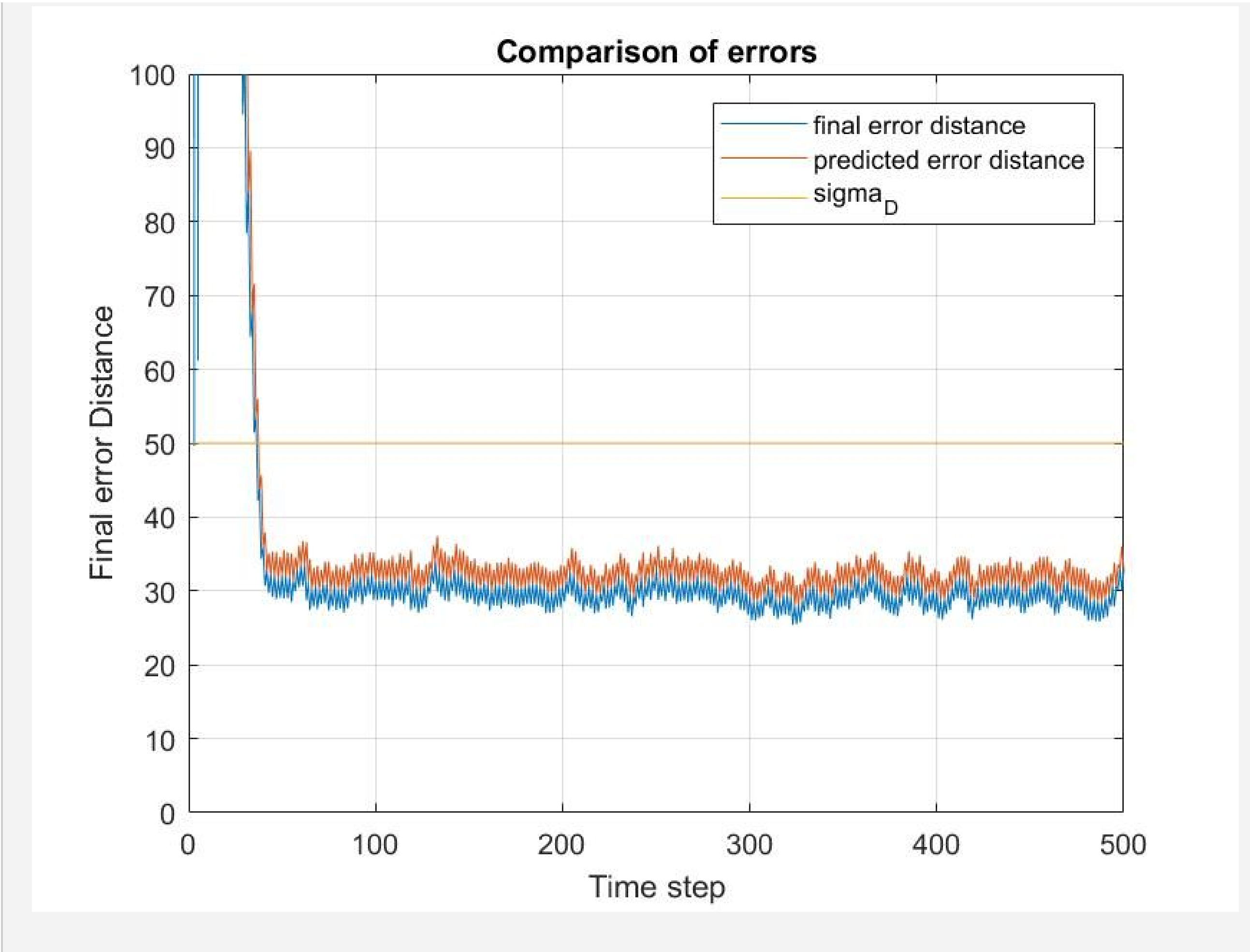



```
% final error
% generation of M=500 realiztions of trajectories

px = nan(1,N);
py = nan(1,N);

for i=1:(N-1)
    px(i) = sqrt(Pfl{i}(1,1));
    py(i) = sqrt(Pfl{i}(3,3));
end



M=500;
b = cell(1,M);
D = cell(1,M);
bm = cell(1,M);
Dm = cell(1,M);
for i=1:M
    [~,~, b{i},D{i}, bm{i},Dm{i}] = trajgen_acc(sigmaA, N, T, x1,y1, vx1, vy1,
sigmab1,sigmab2,sigmaD);
end
```

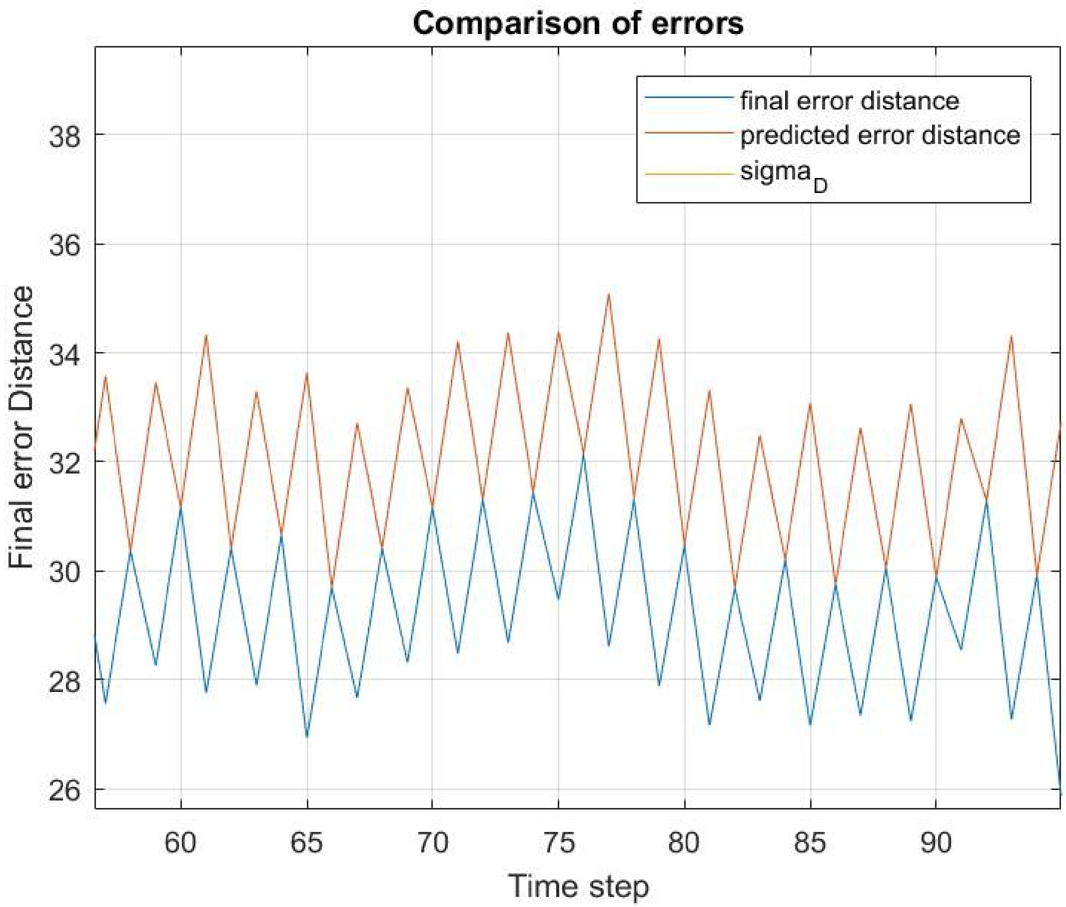
```
% Kalman-filtration of generated trajectories
bfl = cell(1,M);
Dfl = cell(1,M);
bpr = cell(1,M);
Dpr = cell(1,M);

Xfl_ = cell(1,M);
Xpr_ = cell(1,M);
xfl = cell(1,M);
yfl = cell(1,M);
xpr = cell(1,M);
ypr = cell(1,M);

for i=1:M
    [Xpr_{i},~,Xfl_{i},~,~] = extended_kalman_filter(X0,P0,F,Q,R1,R2,bm{i},Dm{i});
    xfl{i} = Xfl_{i}(1,:);
    yfl{i} = Xfl_{i}(3,:);
    [Dfl{i}, bfl{i}] = cartesian2polar(xfl{i},yfl{i});

    xpr{i} = Xpr_{i}(1,:);
    ypr{i} = Xpr_{i}(3,:);
    [Dpr{i}, bpr{i}] = cartesian2polar(xpr{i},ypr{i});
end

fleb = final_error(bfl, b);
fleD = final_error(Dfl, D);

preb = final_error(bpr, b);
preD = final_error(Dpr, D);

figure
plot(t,fleb, t,preb, t,sigmab1*ones(1,N));
legend('final error betta', 'prediction error betta', 'sigma_b');
ylabel('Final error Betta')
xlabel('Time step')
title('Comparison of errors')
ylim([0,0.01])
grid on;
```



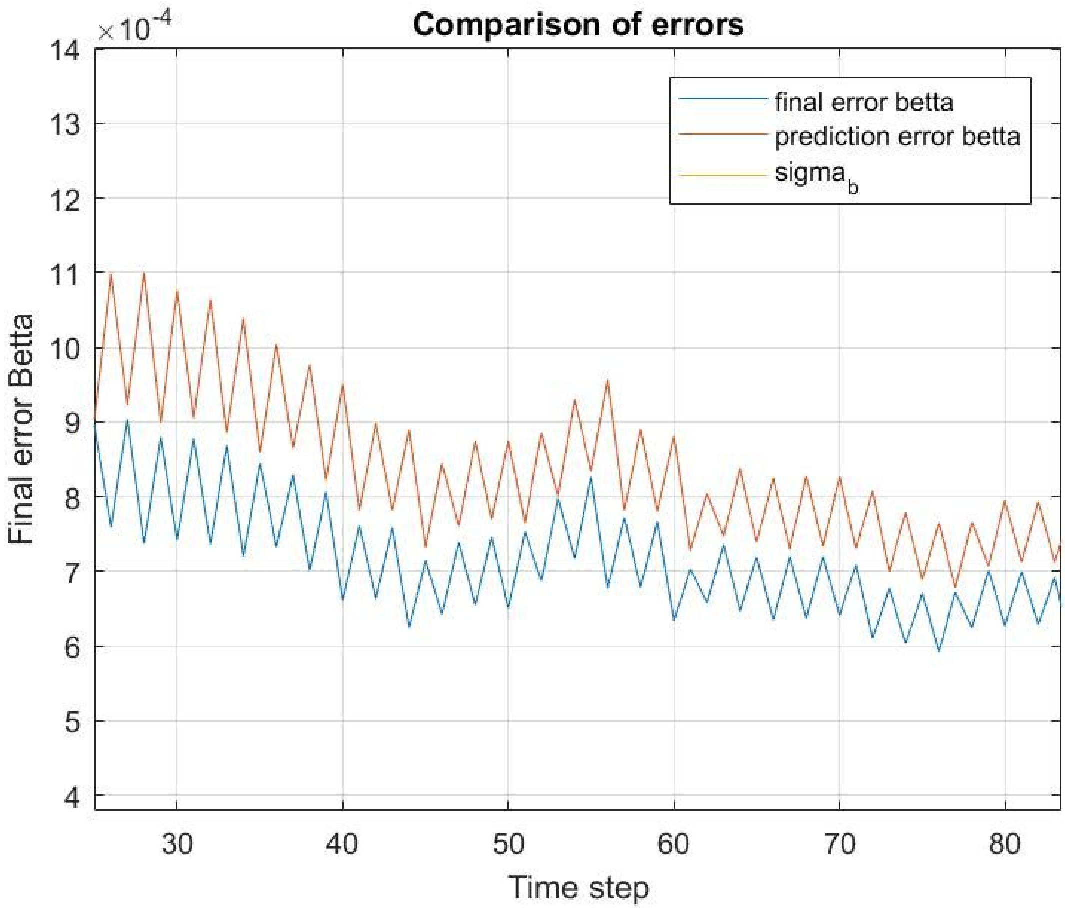

```
figure
plot(t,fleD, t,preD, t,sigmaD*ones(1,N));
legend('final error distance', 'predicted error distance', 'sigma_D');
ylabel('Final error Distance')
xlabel('Time step')
title('Comparison of errors')
ylim([0,100])
grid on;
```



Comparison of errors
final error distance
predicted error distance
sigma_D
Final error Distance
Time step
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500



We have data for angle for both even and odd kinds of steps. However, we have only measurements of range for odd steps. The filtration error increase, when we just have measurements of angle and don’t have measurements of range. The graph illustrates this phenomenon.



The error of estimation decreases on even steps, as we have additional data of angle at these moments of time.

On both graphs we can see that filtration errors are less than extrapolated ones.

Published with MATLAB® R2017b