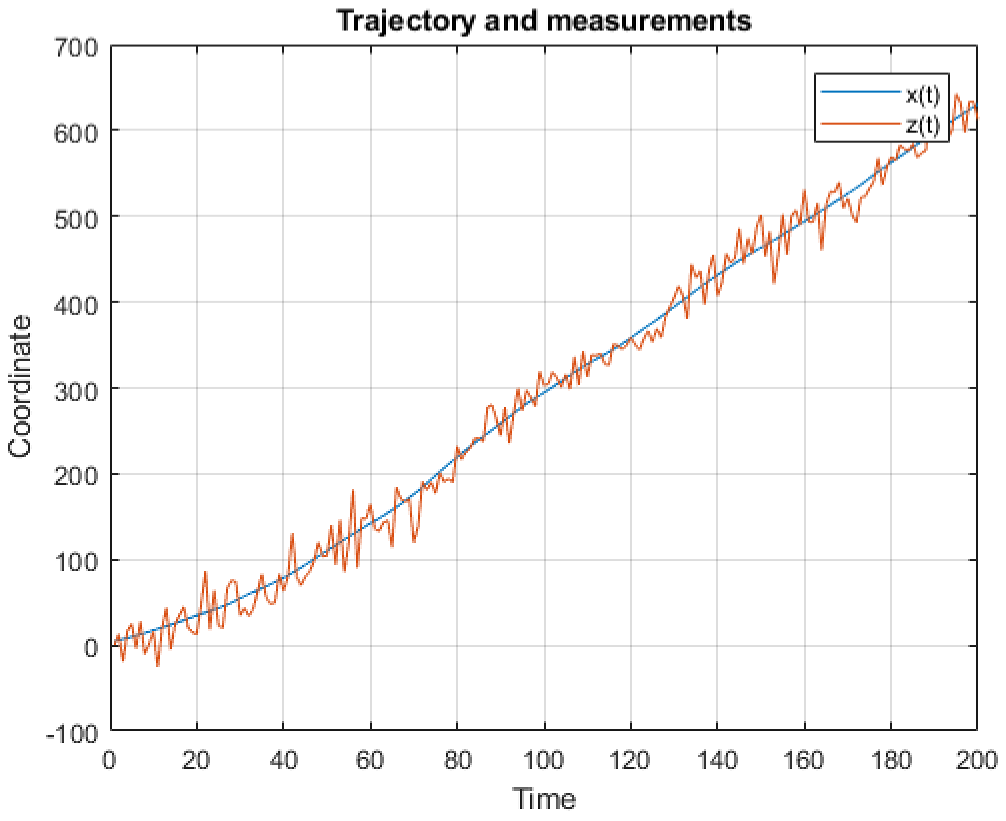
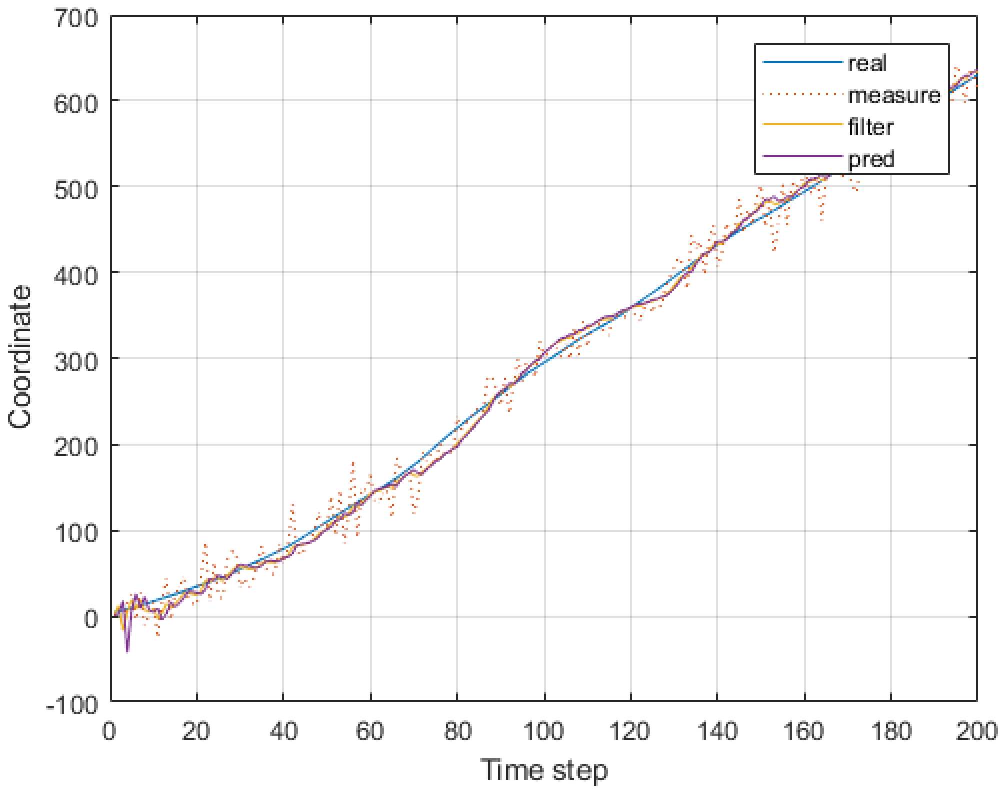
Table of Contents

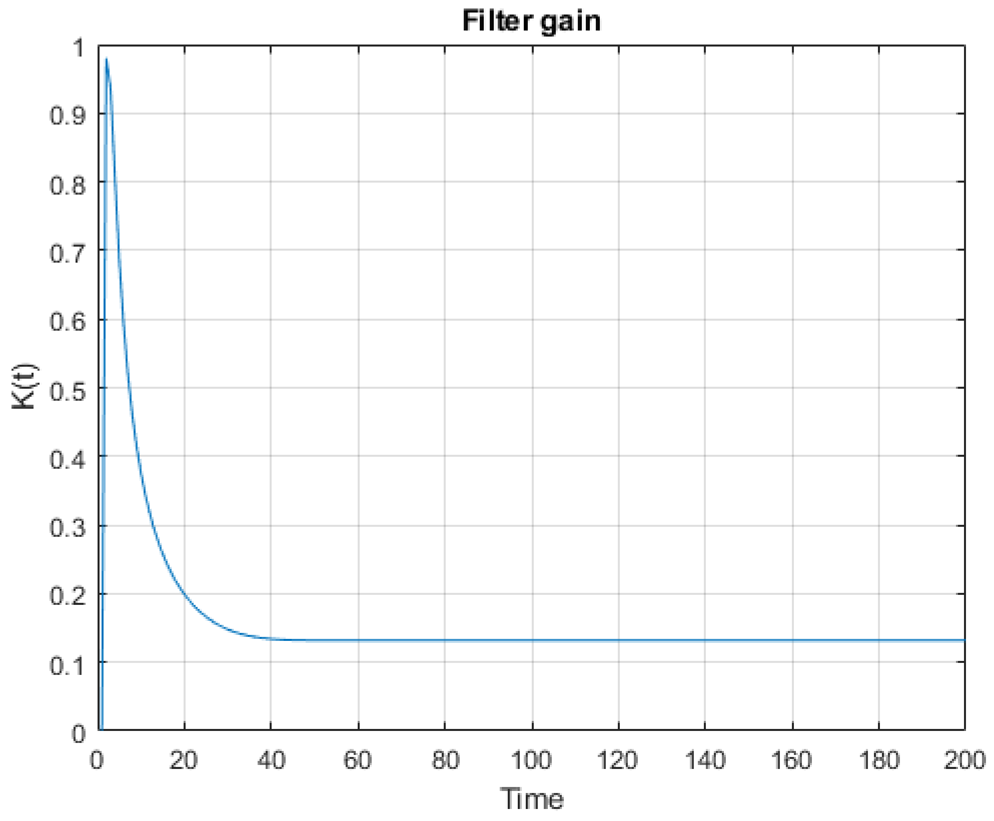
```
% trajectory generation
close all;
clear;
N = 200;
T=1;
v1=1;
sigmaA=0.2;
sigmaN=20;
x1=5;
[x, z] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);

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;

[Xpr,Ppr,Xfl,Pfl,K] = kalman_filter(X0,P0,F,Q,H,R,z);

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

figure(3)
plot(t,K(1,:));
xlabel('Time');
ylabel('K(t)');
title('Filter gain');
grid on;
```

```
% standart deviation of estimation error
p = nan(1,N);
for i=1:(N-1)
    p(i) = sqrt(Pfl{i}(1,1));
end
figure(4)
plot(t,p);
ylabel('sqrt(P_f_i_l_t_e_r(1,1))')
xlabel('Time')
title('Standart deviation of estimation error');
grid on;

% extrapolation on m=7 step on every time step
% graphs show that prediction to several number of steps is lower in
% quality that one for the one future step
m = 7;
[Xprm,Pprm,Xflm,Pflm,Km] = kalman_filter_extra(X0,P0,F,Q,H,R,z,m);

figure(5)
plot(t,x, t,z,':', t,Xflm(1,:), t,Xprm(1,:));
legend('real', 'measure', 'filter', 'pred');
ylabel('Coordinate')
xlabel('Time step')
title('Extra filtration')
grid on;

% generation of M=500 realiztions of trajectories
M=500;
X = cell(1,M);
Z = cell(1,M);
for i=1:M
    [X{i}, Z{i}] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
end
% Kalman-filtration of generated trajectories
Xfl_ = cell(1,M);
Xfl_ex = cell(1,M);
xfl = cell(1,M);
xfl_ex = cell(1,M);

for i=1:M
    [~,~,Xfl_{i},~,~] = kalman_filter(X0,P0,F,Q,H,R,Z{i});
    xfl{i} = Xfl_{i}(1,:);
    [~,~,Xfl_ex{i},~,~] = kalman_filter_extra(X0,P0,F,Q,H,R,Z{i},m);
    xfl_ex{i} = Xfl_ex{i}(1,:);
end

fe = final_error(xfl, X);
fem = final_error(xfl_ex, X);

figure(6)
plot(t,fe, t,fem, t,sigmaN*ones(1,N), t,p);
legend('1 filt step', '7 filt steps', 'sigmaN', 'standart deviation');
ylabel('Final error')
```

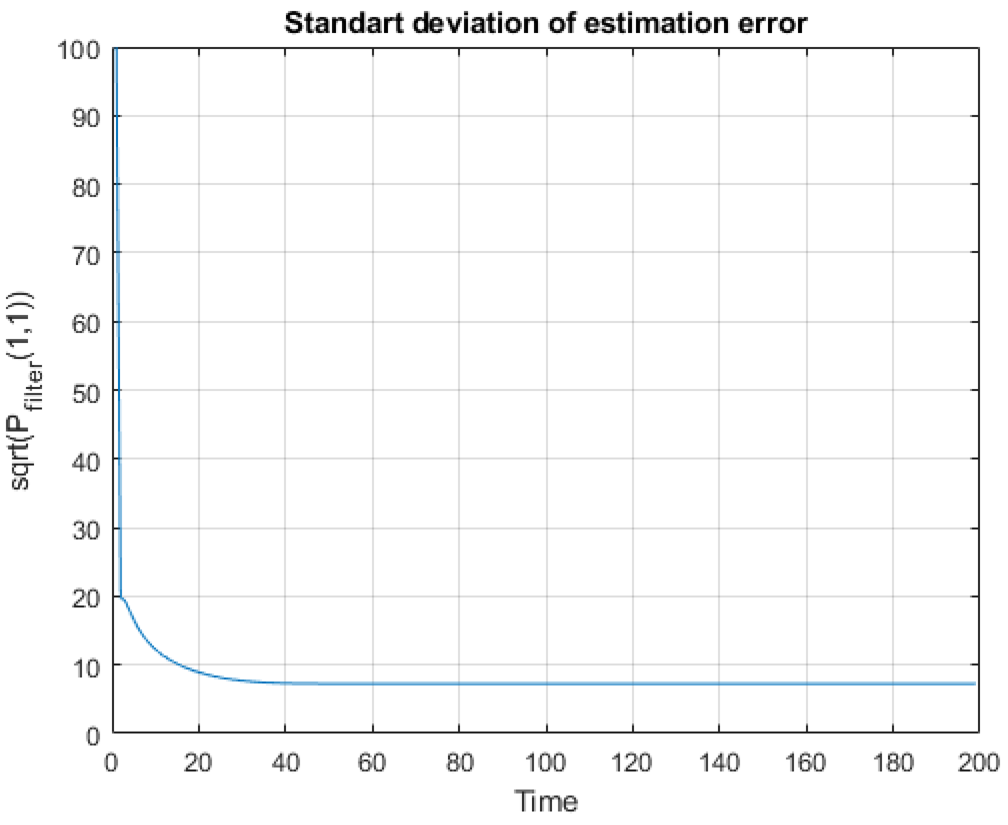
```
xlabel('Time step')
title('Comparison of errors')
grid on;
ylim([0,30]);
% estimation of final error approaches standart deviations error
% filtraion is effective

% more accurate initial conditions
Xfl_1 = cell(1,M);
xfl1 = cell(1,M);
P1 = [100 0; 0 100];
for i=1:M
    [~,~,Xfl_1{i},~,~] = kalman_filter(X0,P1,F,Q,H,R,Z{i});
    xfl1{i} = Xfl_1{i}(1,:);
end

fe1 = final_error(xfl1, X);
figure(7)
plot(t,fe, t,fe1, t,p);
legend('P0=10 000', 'P1=100', 'true est error');
ylabel('Final error')
xlabel('Time step')
title('Different initial conditions P0, P1')
grid on;
ylim([0,30]);
% initial conditions affect only first steps of iterations in
 filtration:
% filter quickly approaches to real trajectory if initial guess is
 better
% Since t=40, the choice of initial conditions doesn't affect future
% observations and estimation of the trajectory.
```


Trajectory and measurements
x(t)
z(t)
Coordinate
Time
700
600
500
400
300
200
100
0
-100
0
20
40
60
80
100
120
140
160
180
200

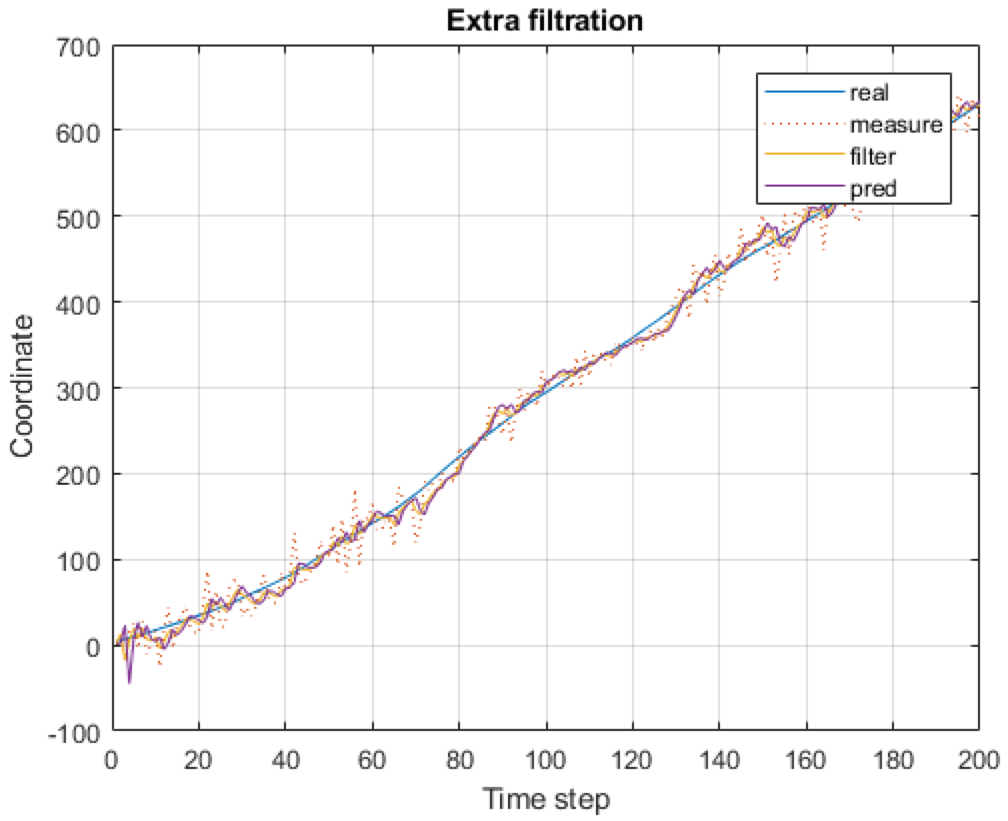

real
measure
filter
pred
Coordinate
Time step
700
600
500
400
300
200
100
0
-100
0
20
40
60
80
100
120
140
160
180
200

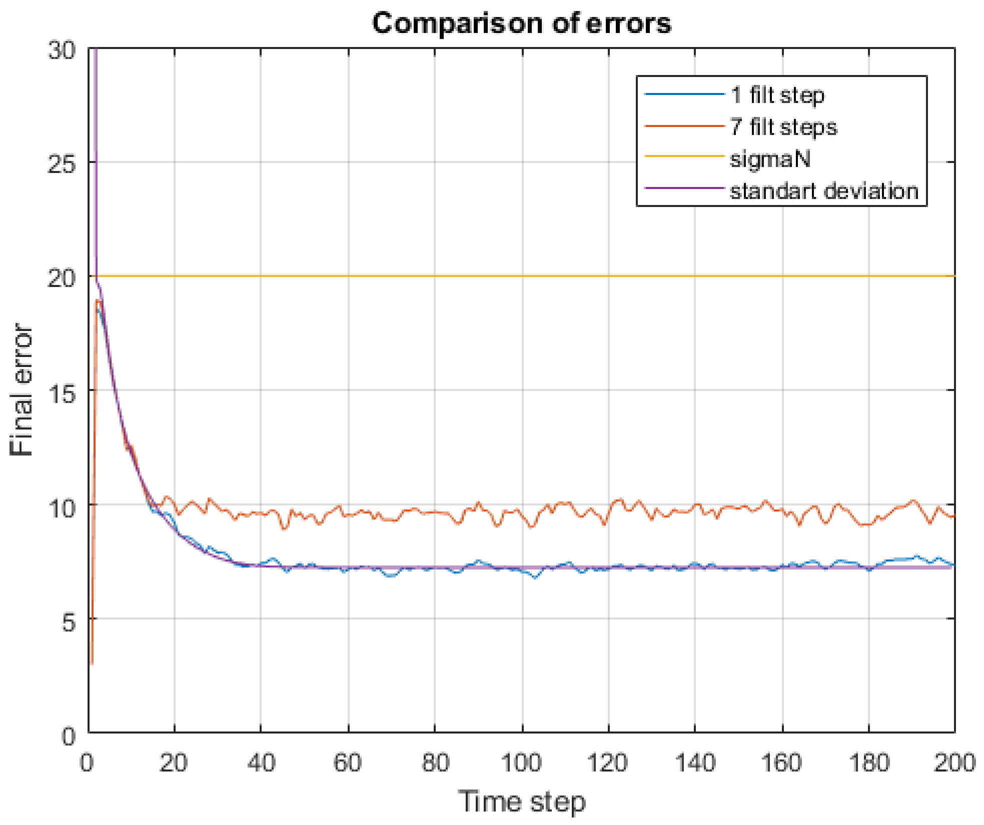


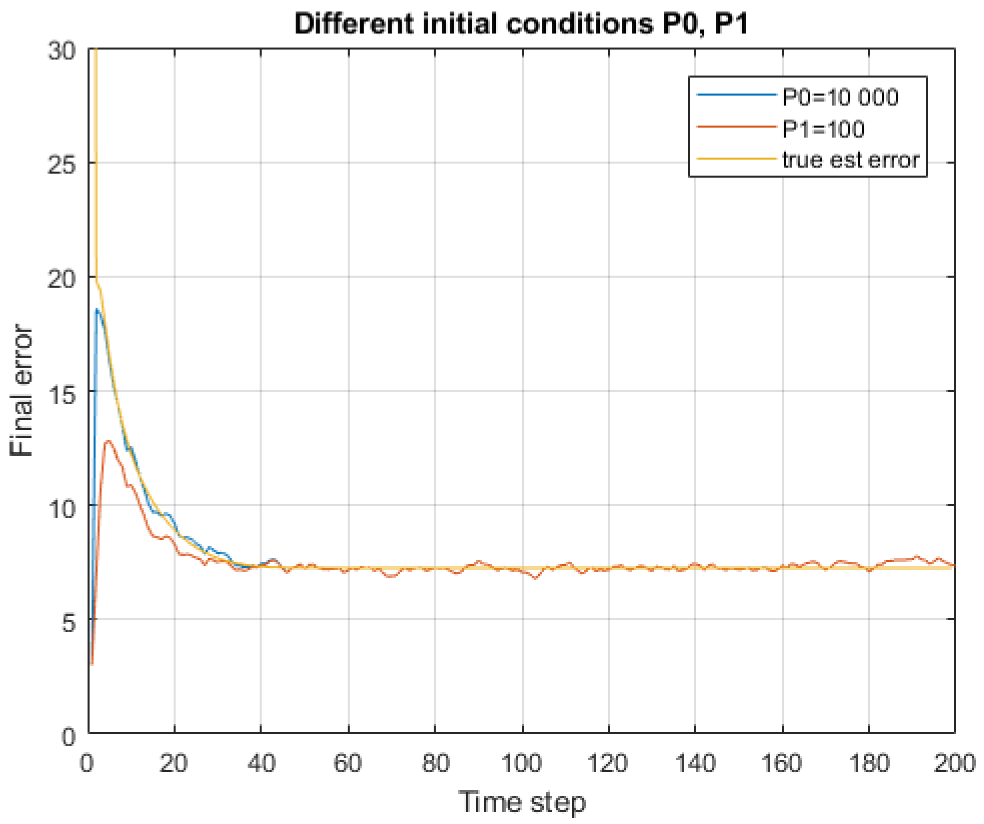
Filter gain
K(t)
Time
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120
140
160
180
200



Standart deviation of estimation error
$\mathrm{sqrt}(P_{filter}(1,1))$
Time
0
10
20
30
40
50
60
70
80
90
100
0
20
40
60
80
100
120
140
160
180
200


Extra filtration
real
measure
filter
pred
Coordinate
Time step

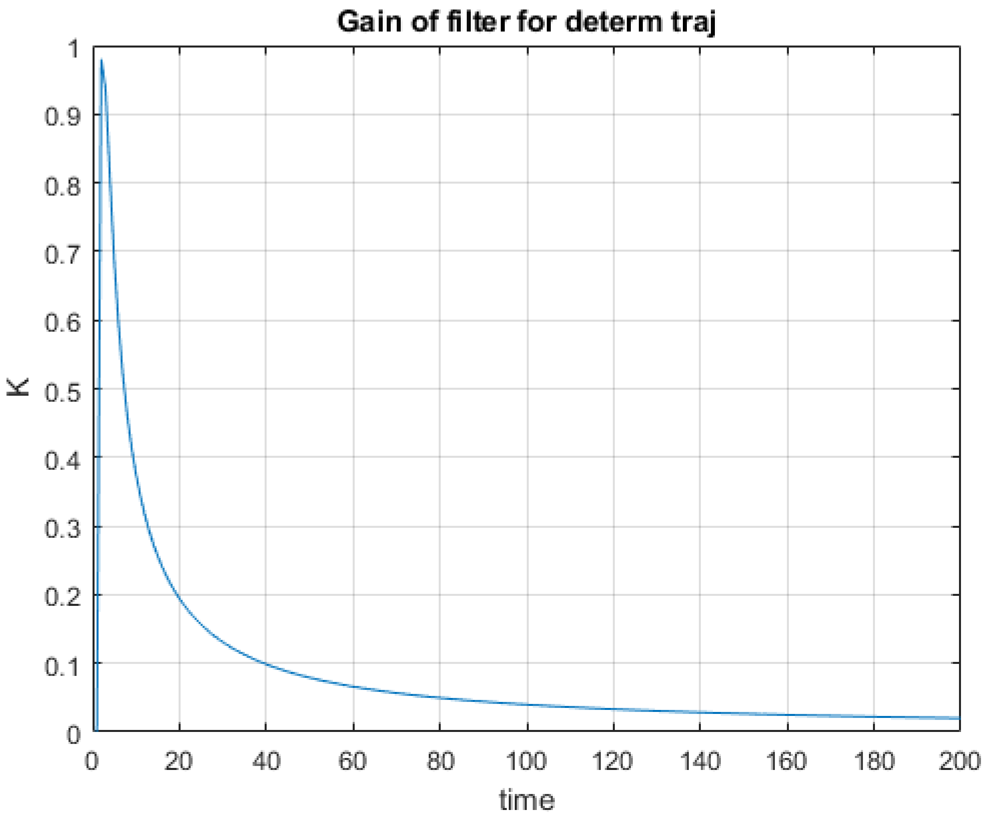

Comparison of errors
1 filt step
7 filt steps
sigmaN
standart deviation
Final error
Time step

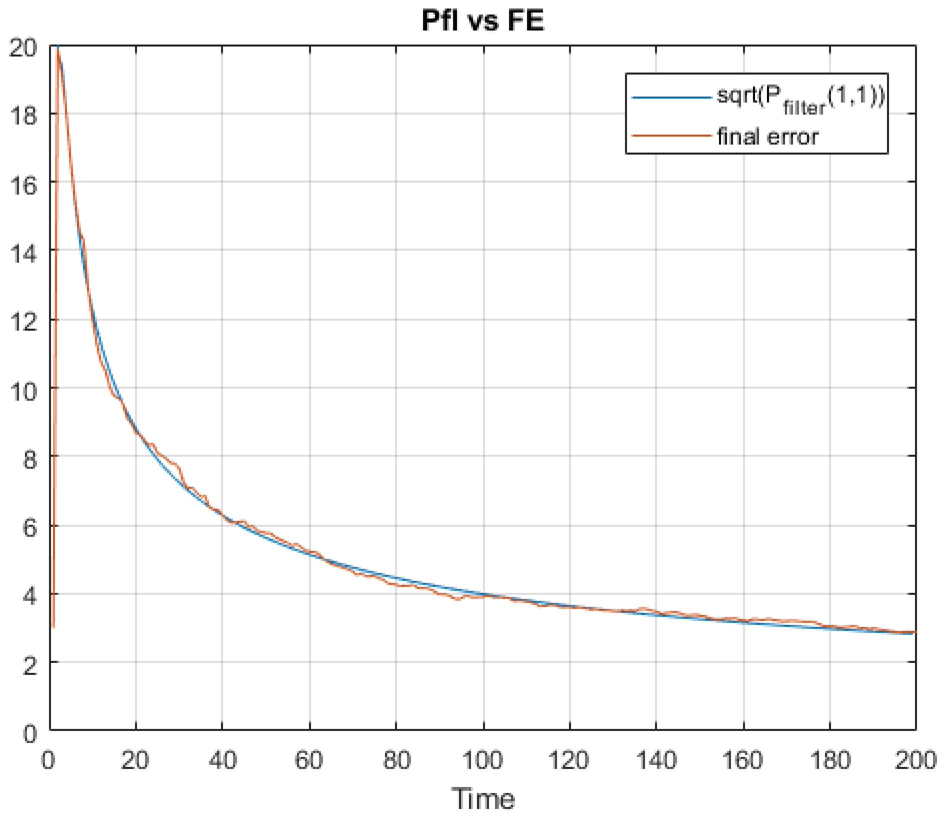


determenistic trajectory

```
sigmaA = 0;
N = 200;
T=1;
v1=1;
sigmaN=20;
Q = zeros(2,2);
M=500;
% generations
X = cell(1,M);
Z = cell(1,M);
for i=1:M
    [X{i}, Z{i}] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
end
% filtrations
Xfl_ = cell(1,M);
xfl = cell(1,M);

for i=1:M
    [~,~,Xfl_{i},~,~] = kalman_filter(X0,P0,F,Q,H,R,Z{i});
    xfl{i} = Xfl_{i}(1,:);
end
[~,~,~,Pfl,k] = kalman_filter(X0,P0,F,zeros(2,2),H,R,Z{1});
```

```
figure(8)
plot(t,k(1,:));
grid on;
xlabel('time');
ylabel('K');
title('Gain of filter for determ traj')
% K->0 for determenistic trajectory as t->InF

p = nan(1,N);
for i=1:(N-1)
    p(i) = sqrt(Pfl{i}(1,1));
end

fe = final_error(xfl, X);

figure(9)
plot(t,p, t,fe);
legend('sqrt(P_f_i_l_t_e_r(1,1))', 'final error')
xlabel('Time')
title('Pfl vs FE');
ylim([0,20]);
grid on;
```



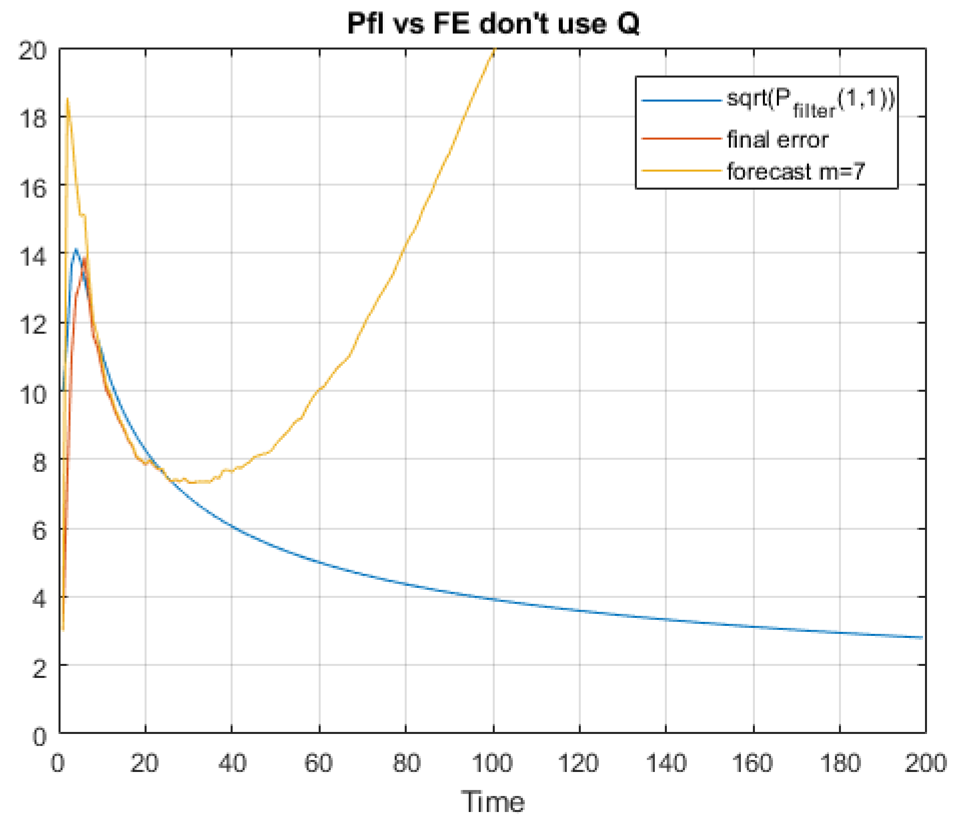


deterministic model of motion disturbed by random acceleration

initialization

```
close all
clear
N = 200;
T=1;
v1=1;
x1=5;
sigmaA=0.2;
sigmaN=20;
Q = zeros(2,2);
M=500;
m=7;
X0 = [2;0];
P0 = [100  0; 0 100];
[F,G,H] = state_space(T);
R = sigmaN^2;
t = 1:N;
% generations
X = cell(1,M);
Z = cell(1,M);
for i=1:M
```

```
    [X{i}, Z{i}] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
end
% filtrations
Xfl_ = cell(1,M);
xfl = cell(1,M);

for i=1:M
    [~,~,Xfl_{i},~,~] = kalman_filter(X0,P0,F,Q,H,R,Z{i});
    xfl{i} = Xfl_{i}(1,:);
    [~,~,Xfl_ex{i},~,~] = kalman_filter_extra(X0,P0,F,Q,H,R,Z{i},m);
    xfl_ex{i} = Xfl_ex{i}(1,:);
end
[~,~,~,Pfl,k] = kalman_filter(X0,P0,F,zeros(2,2),H,R,Z{1});

p = nan(1,N);
for i=1:(N-1)
    p(i) = sqrt(Pfl{i}(1,1));
end

fe = final_error(xfl, X);
fem = final_error(xfl_ex, X);

figure(10)
plot(t,p, t,fe, t,fem);
legend('sqrt(P_f_i_l_t_e_r(1,1))', 'final error', 'forecast m=7')
xlabel('Time')
title('Pfl vs FE don''t use Q');
ylim([0,20]);
grid on;
% error quickly increases if we don't take into account random
 acceleration
% in trajectory.
```

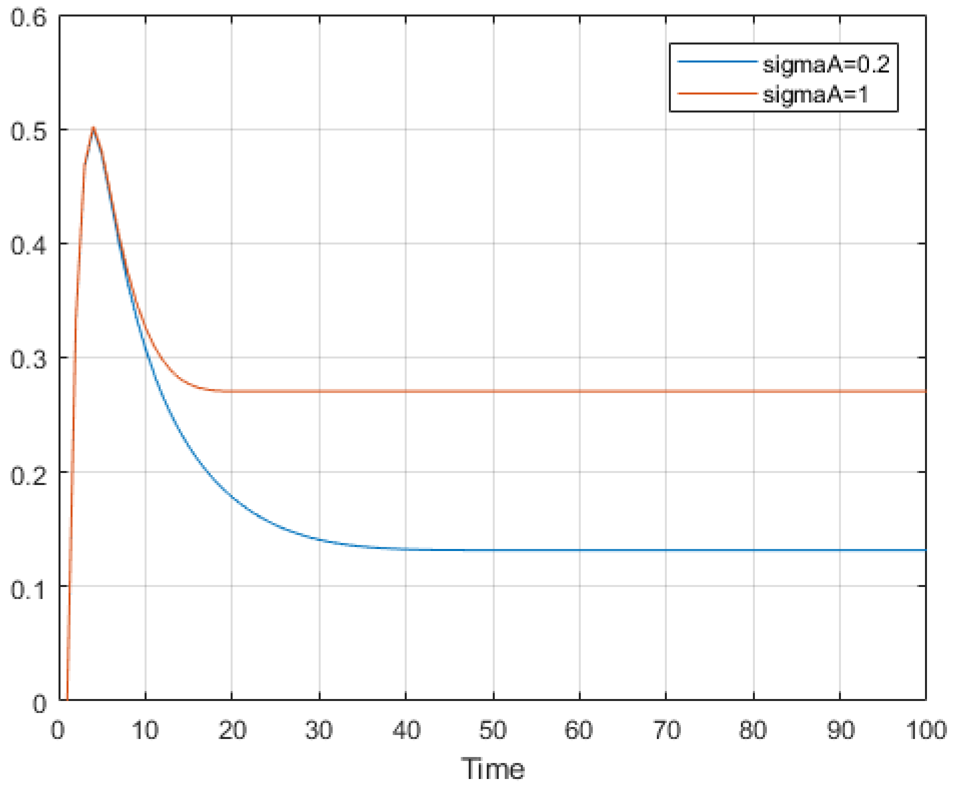


comparison of trajectories with different uncertainty

initialization

```
close all
clear
N = 200;
T=1;
v1=1;
x1=5;
sigmaA1=0.2;
sigmaN=20;
sigmaA2=1;
X0 = [2;0];
P0 = [100  0; 0 100];
[F,G,H] = state_space(T);
Q1 = G*G'*sigmaA1^2;
Q2 = G*G'*sigmaA2^2;
R = sigmaN^2;
t = 1:N;

% generations
[~, z1] = trajgen_acc(x1, sigmaN, sigmaA1, N, T, v1);
[~, z2] = trajgen_acc(x1, sigmaN, sigmaA2, N, T, v1);
```

```
% filtrations
[~,~,~,~,k1] = kalman_filter(X0,P0,F,Q1,H,R,z1);

[~,~,~,~,k2] = kalman_filter(X0,P0,F,Q2,H,R,z1);

figure(1)
plot(t,k1(1,:), t,k2(1,:));
grid on
xlim([0,100])
legend('sigmaA=0.2', 'sigmaA=1')
xlabel('Time')
% less the uncertainty in acceleration more diffucult to detect.
% Therefore filter gain K->const slowlier.
```



sensitivity of filter to underestimated non-optimal filter gain

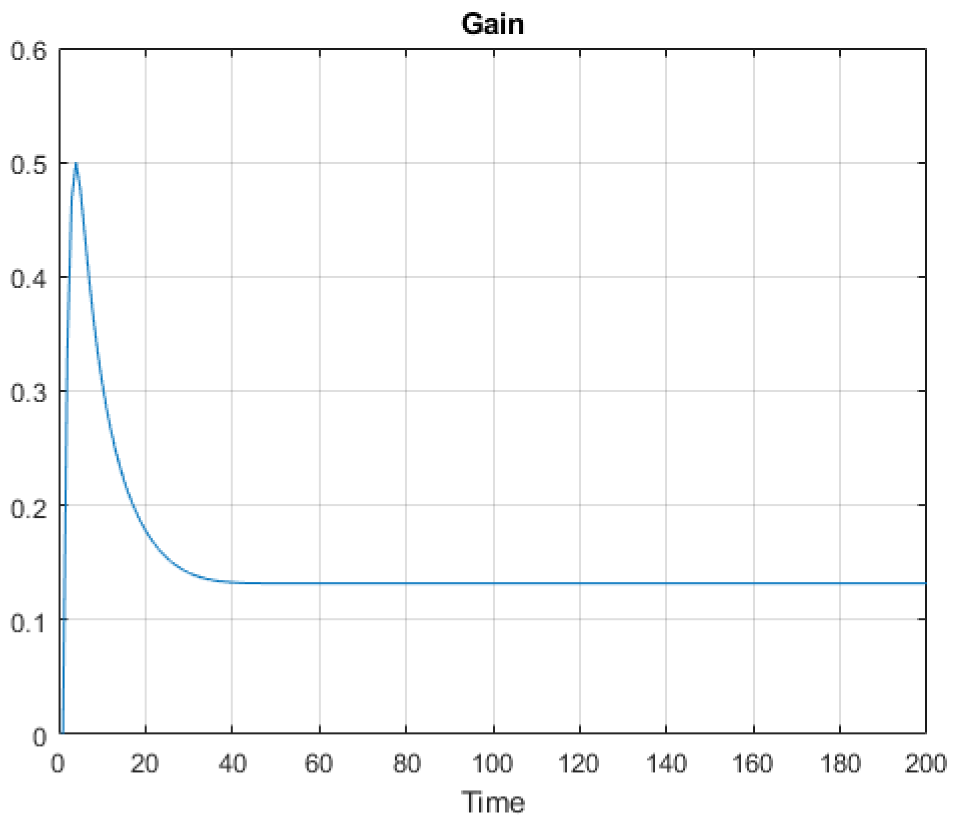
initialization

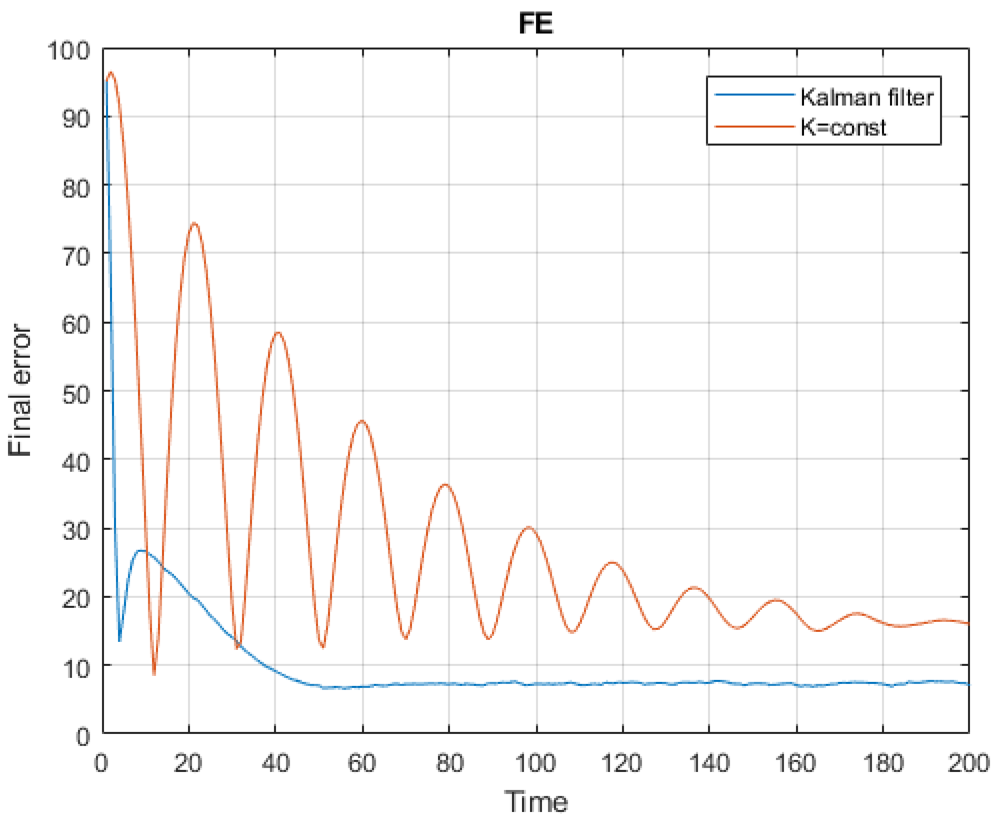
```
close all
clear
N = 200;
M = 500;
T=1;
```

```
v1=1;
x1=5;
sigmaA=0.2;
sigmaN=20;
X0 = [100; 5];
P0 = [100  0; 0 100];
[F,G,H] = state_space(T);
Q = G*G'*sigmaA^2;
R = sigmaN^2;
t = 1:N;

[~, z] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
[~,~,~,~,K] = kalman_filter(X0,P0,F,Q,H,R,z);

k = K(1,200);
figure(2)
plot(t,K(1,:));
xlabel('Time')
title('Gain');
grid on;

display(strcat('Optimal filter gain: ',num2str(k)));

X = cell(1,M);
Z = cell(1,M);
for i=1:M
    [X{i}, Z{i}] = trajgen_acc(x1, sigmaN, sigmaA, N, T, v1);
end
% filtrations
Xfl_ = cell(1,M);
Xfl_const = cell(1,M);
xfl = cell(1,M);
xfl_const = cell(1,M);
for i=1:M
    [~,~,Xfl_{i},~,~] = kalman_filter(X0,P0,F,Q,H,R,Z{i});
    xfl{i} = Xfl_{i}(1,:);
    [~,~,Xfl_const{i},~] =
 kalman_filter_const_gain(X0,P0,F,Q,H,R,Z{i},k/5);
    xfl_const{i} = Xfl_const{i}(1,:);
end

fe = final_error(xfl, X);
fe_const = final_error(xfl_const, X);

figure(1)
plot(t,fe, t,fe_const);
legend('Kalman filter','K=const');
xlabel('Time')
ylabel('Final error')
title('FE');
grid on;
% if filter gain k=const is underestimated and non-optimal, then there
 are
```

```
% oscilations in final error function, depending on the magnitude of
 k.
% Stable error value is higher.
```

```
Optimal filter gain:0.13185
```





Published with MATLAB® R2017b