A Preliminary Application of Kalman Filter

= =!

System Information

Consider, under the linear-Gaussian assumption, the tracking of a target that is a nearly-constant-velocity planar motion using position-only measurements, given by

$$x_k = F_{k-1}x_{k-1} + \Gamma_{k-1}w_{k-1} \tag{1a}$$
$$z_k = H_k x_k + v_k \tag{1b}$$

with

$$F_k = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Gamma_k = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}, \quad H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$
$$\bar{w}_k = 0, \quad Q_k = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}, \quad \bar{v}_k = 0, \quad R_k = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} = I \tag{2}$$

where $T = 1$ is the sampling interval, x is the state of the target and defined as $x = [x, \dot{x}, y, \dot{y}]'$ with $x_0 = [120, 4, 25, 0]'$, with (x, y) denoting the coordinates in the x-y plane. The Kalman filter for this problem is stable because the system is observable and, if $q \neq 0$, controllable.

Initialization information:

$$\hat{x}_0 = [125, 4.1, 26, 0.1]', \quad P_0 = 0.2I_{4\times4} \tag{3}$$

Basic Kalman Filter

For the above linear-Gaussian system, the basic Kalman Filter scheme can be outlined below:

1. Initialization: $\hat{x}_0$, P_0;
2. Prediction:

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} + \Gamma_{k-1}\bar{w}_{k-1} \tag{4a}$$
$$\hat{z}_{k|k-1} = H_k\hat{x}_{k|k-1} + \bar{v}_k \tag{4b}$$
$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F'_{k-1} + \Gamma_{k-1}Q_{k-1}\Gamma'_{k-1} \tag{4c}$$
$$S_k = H_k P_{k|k-1}H'_k + R_k \tag{4d}$$
$$K_k = P_{k|k-1}H'_k S_k^{-1} \tag{4e}$$

3. Update: given observation $z_k = H_k x_k + v_k$ at time k,

$$\tilde{z}_k = z_k - \hat{z}_{k|k-1} \tag{5a}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z}_k \tag{5b}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k' \tag{5c}$$

4. Return to step 2. and perform the next recursion.

Based on the scheme of basic Kalman Filter, the following computer simulations were performed.

Case 1: $q = 0$

When $q = 0$, the process noise has zero expectation with zero variance, that is, there is no process noise in the system at all. Therefore, the system states transition formula turns into

$$x_k = F_{k-1} x_{k-1}.$$

As we can see, in this case the target will move along the X direction with a constant speed of 4 per time step, besides, the y coordinate will remain unchanged. Figure **??** shows in a Monte Carlo run, the target moves as we expected.

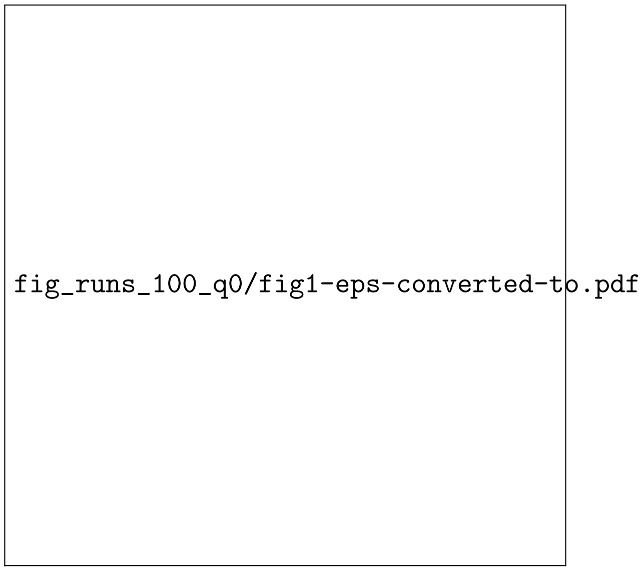


Figure 1: Planar motion of the tracking target

And taking more attention at Figure **??**, and comparing the real value of the system state and the estimated ones, one can discover that after a certain period deviation from the real state, the estimate finally approximating the real state, though in Figure **??** this trend is not much apparent.

Figure **??** and Figure **??** reveal more details about the system state's evolution. As time goes on, the estimate values are approaching the real states, while the measurements fluctuate over the real(expected) values, both the position plots and velocity plots tell the same story here.

fig_runs_100_q0/fig2-eps-converted-to.pdf

(a) X direction component of position

fig_runs_100_q0/fig4-eps-converted-to.pdf

(b) X direction component of velocity

Figure 2: Motion in the x-coordinate.

fig_runs_100_q0/fig3-eps-converted-to.pdf

(a) Y direction component of position

fig_runs_100_q0/fig5-eps-converted-to.pdf

(b) Y direction component of velocity

Figure 3: Motion in the y-coordinate.

Finally look at the Figure **??**, from this figure we can gain deeper perception about the Kalman Filter. Why those estimation of system state will approximate the real value? Because the variance bewteen real state and estimated ones goes down, asymptoticly matching the standard deviation. Though the measurement is contaminated by measurement noise, every time the measurement runs in, the knowledge about the real state increases, in other words, the uncertainty decreases. That is the beauty of Kalman Filter, quit in a natural way.

fig_runs_100_q0/fig6-eps-converted-to.pdf

(a) RMS-position and standard deviation

fig_runs_100_q0/fig7-eps-converted-to.pdf

(b) RMS-velocity and standard deviation

Figure 4: Comparison between RMS and standard deviation.

Case 2: $q \neq 0$

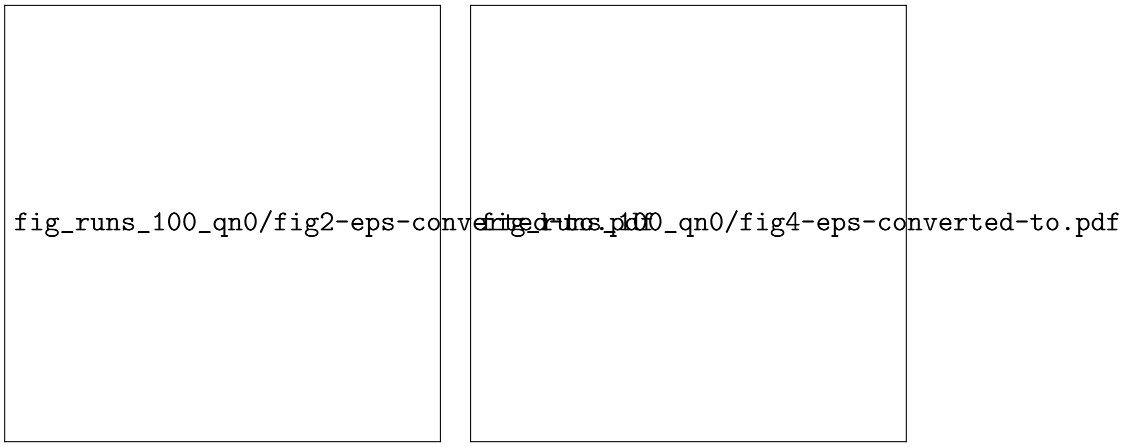
Then we come into the case that $q \neq 0$, which means, from the former analysis, the process noise is not zero, and it can affect the behavior of the system.

fig_runs_100_qn0/fig1-eps-converted-to.pdf

Figure 5: Planar motion of the tracking target

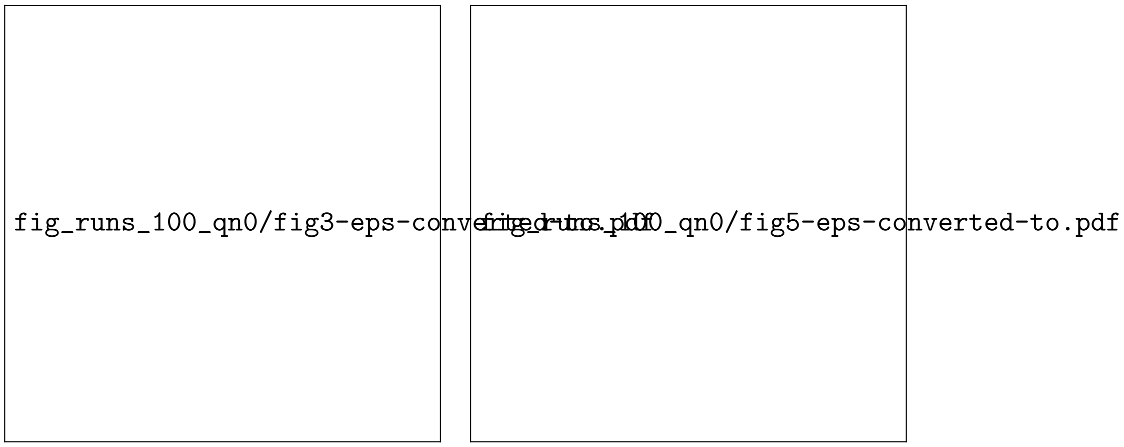
Due to the process noise, in the y direction, the position no more stays the initial value, so does the velocity in y direction(Figure **??** and Figure **??**, Figure **??**). The process noise is actually a kind of input that drives the system to evolution. But the existence of input does not interfere the performence of Kalman Filter, as we can see from Figure **??** that the estimate value is close to the real value. And impose a comparison between Figure **??** and Figure **??** we may find that when $q \neq 0$, the approximation to the real value seems faster.

Apart from the above discussion, in Figure **??** and Figure **??**, there is some delay between the fluctuation of real value and the fluactuation of estimate value. This feature is not explicitly revealed in the filter's scheme.


fig_runs_100_qn0/fig2-eps-converted-to.pdf
fig_runs_100_qn0/fig4-eps-converted-to.pdf

(a) X direction component of position (b) X direction component of velocity

Figure 6: Motion in the x-coordinate.


fig_runs_100_qn0/fig3-eps-converted-to.pdf
fig_runs_100_qn0/fig5-eps-converted-to.pdf

(a) Y direction component of position (b) Y direction component of velocity

Figure 7: Motion in the y-coordinate.

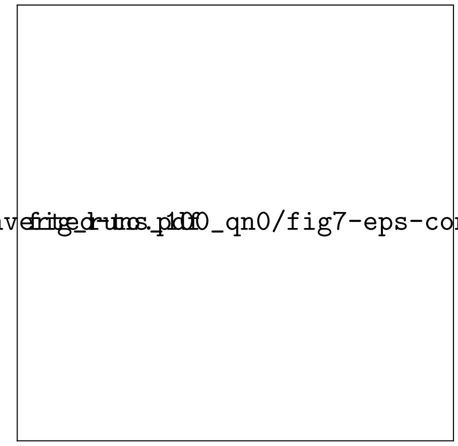
Here comes the final plot of this report. The RMS curve here behaves different from the former case in two way:

1. approximates to the standard deviation faster;

2. oscillates around the standard deviation curve after their intersecting.

Looking back at the Kalman Filter scheme, from Equation 4 and Equation 5 one can derive that:

(a) RMS-position and standard deviation



(b) RMS-velocity and standard deviation

Figure 8: Comparison between RMS and standard deviation.

If $q = 0$, therefore the self-covariance matrix Q_k is zero too. While $q \neq 0$ brings a non-zero self-covariance matrix Q_k. And essentially Q_k is correlated to the decreasing of $P_{k|k}$, in another way, the standard deviation. More specificlly, a larger Q_k coresponds to a sharper decrease of $P_{k|k}$.

That is why when $q \neq 0$, the RMS and state estimation approximates faster than in the case of $q = 0$.

Conclusion

Kalman Filter is a good esitimator for linear-Gaussian system. The uncertainty decreases along the system evolution, at the same time, the estimating accuracy increases.

Comparing the case of $q = 0$ and $q \neq 0$, they differs mainly in the presence of processing noise. And the corresponding effects on system's behavior mainly lie in: a) estimation approximates faster in case of processing noise; b) estimation oscillates around the real value due to the processing noise.

As the footnote of page 179 in the book *Estimation with Applications to Tracking and Navigation* says: noise is beneficial – it lubricates the system. . . More importantly, it was found via psychological experiments that people will lose their sanity without noise, which is particularly true for experts on stochastic(especially those dealing with estimation and tracking).

Appendix – MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% my_kal_1.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
clear all
clc
home
close all
%% parameters
sim_time = 100;
runs = 100;

%%%
T = 1;
q = 0.0;
r = 1;

%%%
m_v_k = [0; 0]; m_omega_k = [0; 0];

F_k = [1, T, 0, 0; 0, 1, 0, 0;
       0, 0, 1, T; 0, 0, 0, 1];

Tao_k = [T^2/2,     0;    T,     0;
         0,     T^2/2;    0,     T;];

H_k = [1, 0, 0, 0; 0, 0, 1, 0];

Q_k = [q, 0; 0, q];

R_k = [r, 0; 0, r];

%% For RMS calculation & comparison
rms_position = zeros(sim_time, 1);
rms_velocity = zeros(sim_time, 1);

%% record useful values
record_P_diag = zeros(sim_time, 4);
record_x = zeros(sim_time, 4);
record_x_estimate = zeros(sim_time, 4);
record_z = zeros(sim_time, 2);
%%
for run = 1:runs
    run
```

```
%% initial state of every run
x_0 = [120, 4, 25, 0]';
P_0 = 0.2*eye(4);
F_km1 = F_k;
Tao_km1 = Tao_k;
Q_km1 = Q_k;
m_omega_km1 = m_omega_k;

estimate_x_0 = [125, 4.1, 26, 0.1]';
estimate_x_km1 = estimate_x_0;
estimate_P_km1 = P_0;
real_x_km1 = x_0;
%% random number generator setting
%rng('shuffle');
%%
for i=1:sim_time

    %%% generate random values of plant noise omega & measurement
    %%% noise v.
    if(q≠0)
        %%% if q≠ 0, that is, the system is observable & controllable
        %%% then Kalman filter is stable, we should use
        %%% Steady-State KF.
        %rng('shuffle');
        chol_Q_k = chol(Q_k);
        omega_km1 = m_omega_km1' + ...
            randn(size(m_omega_km1'))*chol_Q_k;
        omega_km1 = omega_km1';
    else
        omega_km1 = [0; 0];
    end
    chol_R_k = chol(R_k);
    %rng('shuffle');
    v_k = m_v_k' + randn(size(m_v_k'))*chol_R_k;
    v_k = v_k';

    %%% update real x
    real_x_k = F_km1*real_x_km1+Tao_km1*omega_km1;
    %%% generate new measurement
    z_k = H_k*real_x_k + v_k;

    %%% prediction
    predict_x_k_km1 = F_km1*estimate_x_km1 + Tao_km1*m_omega_km1;
    predict_z_k_km1 = H_k*predict_x_k_km1 + m_v_k;
    predict_P_k_km1 = F_km1*estimate_P_km1*(F_km1') + ...
        Tao_km1*Q_km1*(Tao_km1');
    S_k = H_k*predict_P_k_km1*(H_k')+R_k;
    K_k = predict_P_k_km1*(H_k')/S_k;

    %%% update
    z_k_tide = z_k - predict_z_k_km1;
    estimate_x_k = predict_x_k_km1 + K_k*z_k_tide;
    estimate_P_k = predict_P_k_km1 - K_k*S_k*(K_k');
```

```
        %%% obsolete
        m_omega_km1 = m_omega_k;
        estimate_x_km1 = estimate_x_k;
        estimate_P_km1 = estimate_P_k;
        real_x_km1 = real_x_k;

        %%% record useful values for plotting
        record_x_estimate(i, :) = estimate_x_k';
        record_x(i, :) = real_x_k';
        if(run==runs)
            record_z(i, :) = z_k';
            record_P_diag(i, :) = diag(estimate_P_k)';
        end
    end

    %% RMS
    rms_position = rms_position + ...
        (record_x_estimate(:, 1)-record_x(:, 1)).^2 + ...
        (record_x_estimate(:, 3)-record_x(:, 3)).^2;
    rms_velocity = rms_velocity + ...
        (record_x_estimate(:, 2)-record_x(:, 2)).^2 + ...
        (record_x_estimate(:, 4)-record_x(:, 4)).^2;
end

%% RMS
rms_position = sqrt(rms_position/runs);
rms_velocity = sqrt(rms_velocity/runs);

%% plot
plotting_for_a_run

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% plotting_for_a_run.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
figure(1) % position
hold on
xlabel('position x')
ylabel('position y')
plot(record_x_estimate(:,1), record_x_estimate(:, 3), 'b-o', ...
    'linewidth',1)
plot(record_x(:, 1), record_x(:, 3), 'r', 'linewidth', 1)
legend('estimate', 'real')

%%
figure(2) % position in x
hold on
xlabel('time step'), ylabel('position x')
plot(record_z(:, 1), 'color',[0.2, 0.7, 0.1], 'Marker', ...
    'v','linewidth', 1)
plot(record_x_estimate(:, 1), 'b', 'Marker', 'o', 'linewidth', 1)
plot(record_x(:, 1), 'r', 'linewidth',2)
legend('measurement', 'estimate', 'real')

%%
figure(3) % position in y
hold on
xlabel('time step'), ylabel('position y')
plot(record_z(:, 2), 'color',[0.2, 0.7, 0.1], 'Marker', ...
    'v','linewidth', 1)
plot(record_x_estimate(:, 3), 'b', 'Marker', 'o', 'linewidth', 1)
plot(record_x(:, 3), 'r', 'linewidth',2)
legend('measurement', 'estimate', 'real')

%%
figure(4) % velocity in x
hold on
xlabel('time step'), ylabel('velocity x')
tmp = record_x_estimate(:, 2);
plot(tmp, 'b', 'Marker', 'o')
plot(record_x(:, 2), 'r', 'linewidth',1);
if(q==0)
ylim([min(record_x_estimate(:, 2))-0.05, 4.1])
end
legend('estimate', 'real')

%%
figure(5) % velocity in y
hold on
xlabel('time step'), ylabel('velocity y')
tmp = record_x_estimate(:, 4);
plot(tmp, 'b', 'Marker', 'o')
plot(record_x(:, 4), 'r', 'linewidth',1);
legend('estimate', 'real')
```

```
%%
figure(6)
plot(rms_position, 'bo-')
hold on
plot(sqrt(record_P_diag(:,1)+record_P_diag(:,3)),'r', 'linewidth', 2)
xlabel('time step'), ylabel('value'), ylim([0, 2])
legend('RMS position', 'standard deviation')

%%
figure(7)
plot(rms_velocity, 'bo-')
hold on
plot(sqrt(record_P_diag(:,2)+record_P_diag(:,4)),'r', 'linewidth', 2)
xlabel('time step'), ylabel('value'), ylim([-0.03 1])
legend('RMS velocity', 'standard deviation')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```