Solutions for examination in Sensor Fusion, 2022-06-02

1. (a) **3 independent virtual measurements can be constructed.**
   $\bar{y}_{i0} = y_i - y_0 = \frac{1}{c}(\|x - s_i\| - \|x - s_0\|) + e_i - e_0$, $i = 1, \ldots, 3$.
   $$\mathbf{cov}\left(\begin{pmatrix}\bar{y}_{10}\\ \bar{y}_{20}\\ \bar{y}_{30}\end{pmatrix}\right) = \begin{pmatrix}R_1 + R_0 & R_0 & R_0\\ R_0 & R_2 + R_0 & R_0\\ R_0 & R_0 & R_3 + R_0\end{pmatrix}$$

   (b) **(i), (vi)**

   (c) **Increase the process noise Q to make the filter rely more on the measurements.**

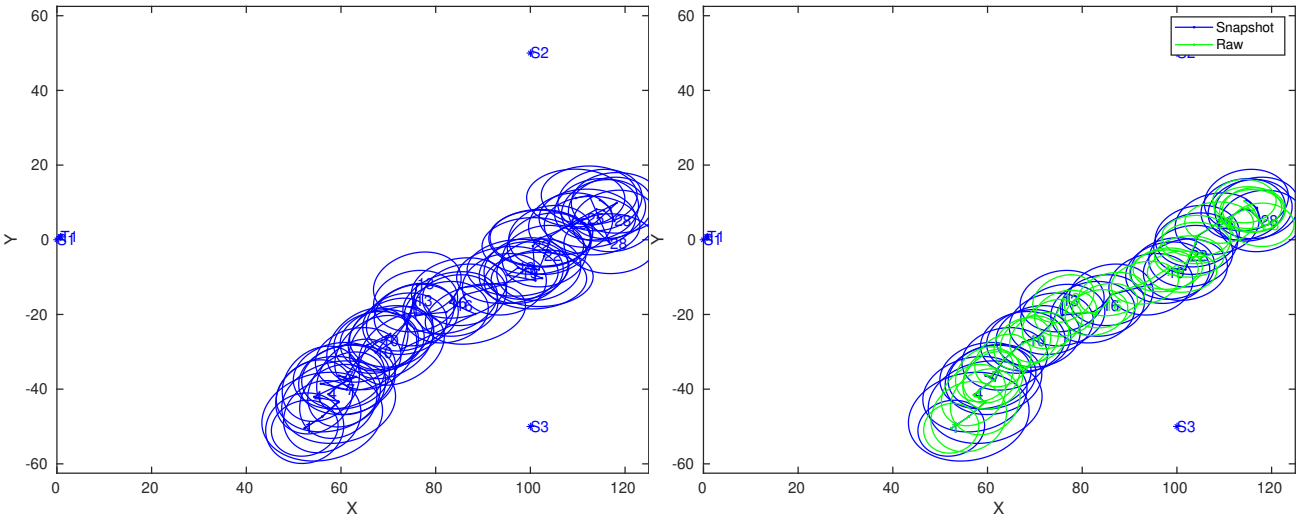
   (d) $\hat{x} = \begin{pmatrix}0.8\\ 2\end{pmatrix}$, $P = \begin{pmatrix}0.2 & 0\\ 0 & 0.33\end{pmatrix}$

   The estimates might be correlated, hence safe fusion should be used.

   (e) **(iv), (v)**

2.
```
%% Exercise 2
clear;
load data20220602
T = mean(diff(ex2_t));  % Sample time

%% Exercise 2a
% Make a TOA measurement model with 3 sensors
sm = exsensor('toa', 3, 1);
% Place the sensors in the correct locations
sm.th = [0 0 100 50 100 -50]';
% Set the measurement noise as given in the exercise
sm.pe = 5^2*eye(3);

% Create combined measurements that are easier to work with
y = [ex2_y1; ex2_y2; ex2_y3]';
Y = sig(y, ex2_t);

% Estimate the trajectory as snapshots
for i = 1:numel(ex2_y1)
    xhat = estimate(sm, Y(i), 'thmask', zeros(6, 1));
    X_snapshot(i, :) = xhat.x0;
    P_snapshot(i, :, :) = cov(xhat.px0);
end
Xhat_snapshot = sig(y, ex2_t, [], X_snapshot, [], P_snapshot);

%Plot the result
figure(1); clf
plot(sm); hold on;
xplot2(Xhat_snapshot, 'conf', 90);

%% Exercise 2b
% We have not much information about the motion, but CV seems reasonable
% given part (a).  The motion model should be the same in both the filter,
% its the same target that we are tracking.  Hence, define it once for all.
mm = exmotion('cv2d', T);
% Set the process noise, this is a tuning parameter
G = kron([T^2/2; T], eye(2));
q = eye(2);
Q = G*q*G';
mm.pv = Q;
% Use the first two measurements for initialization
mm.x0 = [X_snapshot(1, :)'; (X_snapshot(2, :)-X_snapshot(1, :))'/T];
mm.px0 = blkdiag(squeeze(P_snapshot(1, :, :)),...
  10/T*squeeze(P_snapshot(1, :, :)+P_snapshot(2, :, :)));

% Set up the measurement model for the snapshot measurements
sm1 = sensormod(@(t, x, u, th) [eye(2), zeros(2)]*x, [4, 0, 2, 0]);
sm1.pe = 2*squeeze(mean(P_snapshot, 1));  % Pick something reasonable

% Create model for snapshot measurements
mms1 = addsensor(mm, sm1);
Y1 = sig(X_snapshot, ex2_t, [], [], P_snapshot);
```

```
% Run the EKF
Xhat_ekf1 = ekf(mms1, Y1);
xplot2(Xhat_ekf1, 'conf', 90);

% Create model for raw measurements
mms2 = addsensor(mm, sm);
% Run the EKF
Xhat_ekf2 = ekf(mms2, Xhat_snapshot);

figure(2);
plot(sm); hold on;
h = xplot2(Xhat_ekf1, Xhat_ekf2, 'conf', 90);
legend(h, 'Snapshot', 'Raw');

%% Exercise 2c
% Removing one of the sensors results in an abmbiguity in the snapshot
% estimate of the position.  Hence, with additional logics its impossible
% to get reasonable results from (a).  The problem is the same in (b), and
% the solution relying on the snapshot estimates naturally fails too, where
% as given a reasonable initail estimate the filter using the raw data
% should be able to track the object.  This because with a reasonable
% initialization, the EKF will converge to the correct estimate.
```



(a) (b)

Figure 1: Figures for Exercise 2.

3. (a) From the problem description, the hypothesis test becomes

$$\begin{cases} \mathcal{H}_0: & y \sim \mathcal{N}(0,1) \\ \mathcal{H}_1: & y \sim \chi_2^2 = \text{Exp}(2) \end{cases}$$

It comprise two simple hypotheses, hence Neyman-Pearson's lemma gives that the most powerful detector is given by the likelihood-ratio test

$$T(y) = \frac{\text{Exp}(y;2)}{\mathcal{N}(y;0,1)} = \frac{\frac{1}{2}e^{-\frac{1}{2}y}}{\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}y^2}} = \sqrt{\frac{\pi}{2}}e^{-\frac{1}{2}(y-y^2)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma.$$

for $y > 0$ otherwise $T(y) = 0$.
Now, P_{FA} is given by

$$P_{\text{FA}} = P(T > \gamma | \mathcal{H}_0) = \int_{T(y)>\gamma} \mathcal{N}(y;0,1)\, dy,$$

which lacks a simple closed form solution.

Similarly, P_{D} is given by

$$P_{\text{D}} = P(T > T_{\text{th}}|\mathcal{H}_1) = \int_{T(y)>\gamma} \text{Exp}(y; 2)\, dy,$$

which lacks a simple closed form expression.

(b) The CRLB is given by the inverse Fisher information matrix, $P_{\text{CRLB}}(x) = \mathcal{I}(x)$. Given that the noise can be assume Gaussian with covariance matrix $R = \begin{pmatrix} (\sigma^r)^2 & 0 \\ 0 & (\sigma^\phi)^2 \end{pmatrix}$, (4.4) in the textbook provides a suitable expression for

$$\mathcal{I}(x) = \nabla h^T(x) R^{-1} \nabla h(x).$$

It follows from straightforward derivation or by consulting Table 4.1 in the textbook that

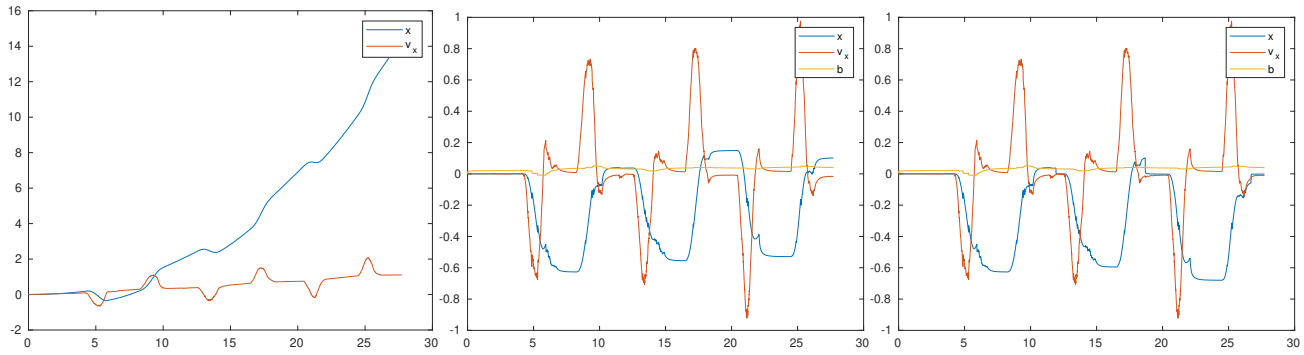
$$\nabla h(x) = \begin{pmatrix} \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|} & \frac{x_k^{\text{y}} - x_0^{\text{y}}}{\|x_k - x_0\|} \\ \frac{-(x_k^{\text{y}} - x_0^{\text{y}})}{\|x_k - x_0\|^2} & \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|^2} \end{pmatrix}.$$

The CRLB can now be computed as

$$P_{\text{CRLB}}(x_k) = \left(\begin{pmatrix} \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|} & \frac{x_k^{\text{y}} - x_0^{\text{y}}}{\|x_k - x_0\|} \\ \frac{-(x_k^{\text{y}} - x_0^{\text{y}})}{\|x_k - x_0\|^2} & \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|^2} \end{pmatrix}^T \begin{pmatrix} (\sigma^r)^{-2} & 0 \\ 0 & (\sigma^\phi)^{-2} \end{pmatrix} \begin{pmatrix} \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|} & \frac{x_k^{\text{y}} - x_0^{\text{y}}}{\|x_k - x_0\|} \\ \frac{-(x_k^{\text{y}} - x_0^{\text{y}})}{\|x_k - x_0\|^2} & \frac{x_k^{\text{x}} - x_0^{\text{x}}}{\|x_k - x_0\|^2} \end{pmatrix} \right)^{-1}$$

4.
```
%% Exercise 4

load data20220602

% Model (p: position, v: velocity, b: bias, u: acceleration)
% input u = acc + bias + w_k:
% p_k+1 = p_k + T*v_k + T^2/2*(u_k-b_k) = p_k + T*v_k - T^2/2*b_k + T^2/2*u_k
% v_k+1 =          v_k +     T*(u_k-b_k) =       v_k + - T*b_k     +     T*u_k
% b_k+1 =                            b_k  =                 b_k
% Define some helper functins to compute transitions matrices based on the
% sample time.
F = @(T)[1, T, -T^2/2;
         0 1 -T;
         0 0 1];
G = @(T)[T^2/2;
         T;
         0];
qa = 0.1^2;  % Reasonable meas error in accelerometer, tuning parameter
qb = 0.0001^2;  % Bias process noise, tuning parameter

x0 = [0; 0; 0];  % Start at standstill, assume 0 bias
P0 = diag([0, 0, 1].^2);  % Start at standstill and uncertain bias

% Extract measurements into more suitable format
t = ex4_t-ex4_t(1);
dT = diff(t);
acc = ex4_acc;

%% a) Dead reckoning, given measurements
% Being a bit lazy, carry around the bias state without using it as it
% simplifies the remaining tasks.
x = x0;
P = P0;
Xhat_a = zeros(2, numel(dT));
for k=1:numel(dT)
  F_ = F(dT(k));
  G_ = G(dT(k));
  x = F_*x + G_*acc(1, k);
  Q = G_*qa*G_'+dT(k)*diag([0, 0, qb]);
  P = F_*P*F_' + Q;
  Xhat_a(:, k) = x(1:2);
end
figure(1); clf
plot(t(1:end-1), Xhat_a');
```

```
legend('x', 'v_x')

%% b) Add a stand still detector and virtual 0 speed measurements
x = x0;
P = P0;
Xhat_b = zeros(3, numel(dT));
Hb = [0 1 0];  % Virtual speed measurement
Rb = 0.04^2;  % How much to trust the virtual meas, tuning parameter
acc_th = 0.1;  % Threshold for stand still detection
for k=1:numel(dT)
  F_ = F(dT(k));
  G_ = G(dT(k));
  x = F_*x + G_*acc(1, k);
  Q = G_*qa*G_'+dT(k)*diag([0, 0, qb]);
  P = F_*P*F_' + Q;

  if norm(acc(1, k)-x(3)) < acc_th  % Stand still detection
    e = 0 - Hb*x;
    S = Hb*P*Hb' + Rb;
    K = P*Hb'/S;
    x = x+K*e;
    P = P - K*Hb*P;
  end

  Xhat_b(:, k) = x;
end

figure(2); clf
plot(t(1:end-1), Xhat_b');
legend('x', 'v_x', 'b')

%% c) Add information about revisiting the starting position
x = x0;
P = P0;
Xhat_c = zeros(3, numel(dT));
Hc = [1 0 0; 0 1 0];  % Virtual measurement, we know 0 pos and 0 speed
Rc = diag([0.01, 0.04].^2);  % Trust in virtual measurement, tuning parameter
for k=1:numel(dT)
  F_ = F(dT(k));
  G_ = G(dT(k));
  x = F_*x + G_*acc(1, k);
  Q = G_*qa*G_'+dT(k)*diag([0, 0, qb]);
  P = F_*P*F_' + Q;

  if norm(acc(1, k)-x(3)) < acc_th  % Stand still detection
    e = 0 - Hb*x;
    S = Hb*P*Hb' + Rb;
    K = P*Hb'/S;
    x = x+K*e;
    P = P - K*Hb*P;
  end
  if k < 10 || k>3000 || k>1340 && k<1350 || k>2100 && k<2110  % at start point
    e = 0 - Hc*x;
    S = Hc*P*Hc' + Rc;
    K = P*Hc'/S;
    x = x+K*e;
    P = P - K*Hc*P;
  end
  Xhat_c(:, k) = x;
end
figure(3); clf
plot(t(1:end-1), Xhat_c');
legend('x', 'v_x', 'b')
```



(a) (b) (c)

Figure 2: Figure for exercise 4.