Solutions for examination in Sensor Fusion, 2023-08-16

1. (a) $\bar{\boldsymbol{y}} = \begin{pmatrix} \|\boldsymbol{p}-\boldsymbol{p^1}\|-\|\boldsymbol{p}-\boldsymbol{p^4}\| \\ \|\boldsymbol{p}-\boldsymbol{p^2}\|-\|\boldsymbol{p}-\boldsymbol{p^4}\| \\ \|\boldsymbol{p}-\boldsymbol{p^3}\|-\|\boldsymbol{p}-\boldsymbol{p^4}\| \end{pmatrix} + \bar{\boldsymbol{e}}$, **where** $\bar{\boldsymbol{e}} \sim \boldsymbol{\mathcal{N}}(\boldsymbol{0}, \bar{\boldsymbol{R}})$ **and** $\bar{\boldsymbol{R}} = \begin{pmatrix} \boldsymbol{R^1}+\boldsymbol{R^4} & \boldsymbol{R^4} & \boldsymbol{R^4} \\ \boldsymbol{R^4} & \boldsymbol{R^2}+\boldsymbol{R^4} & \boldsymbol{R^4} \\ \boldsymbol{R^4} & \boldsymbol{R^4} & \boldsymbol{R^3}+\boldsymbol{R^4} \end{pmatrix}$.

   The key here is to realize that $\mathsf{E}(e^i - e^1)(e^j - e^1) = R^i + R^1$ if $i = j$ and R^1 otherwise.

   (b) $\bar{\boldsymbol{T}}(\boldsymbol{y}) = \boldsymbol{2 \log \sup_x p(y|\mathcal{H}_1)/p(y|\mathcal{H}_0) = y^T H (H^T H)^{-1} H^T y}$

   (c) $\hat{\boldsymbol{x}}^{\mathbf{MV}} = \boldsymbol{0.5}$

   $\hat{x}^{\mathrm{MV}} = \mathsf{E}_{x|y}\, x = 0.25 \cdot (-4) + 0.1 \cdot (-1) + 0.05 \cdot 0 + 0.15 \cdot 1 + 0.1 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 4 + 0.05 \cdot 5 = 0.5$

   (d) $\boldsymbol{p(x^1|x^2 = \hat{x}) = \mathcal{N}(x^1; \mu^1 + R^{12}(R^2)^{-1}(\hat{x} - \mu^2), R^1 - R^{12}(R^2)^{-1}(R^{12})^T)}$

   The given information gives that $\binom{x^1}{x^2} \sim \mathcal{N}\big(\binom{\mu^1}{\mu^2}, \begin{pmatrix} R^1 & R^{12} \\ (R^{12})^T & R^2 \end{pmatrix}\big)$, to which Lemma 7.1 can be applied to get the asked for posterior distribution.

   (e) **Increase Q until a reasonable trajectory is obtained.**

   As R is chosen to match the sensor specification it should probably be left alone. Laggy estimates indicate that we trust the motion model too much, and hence Q should be increased.

2.
```
%% Ex 2
load('data20230816.mat');

X = ex2_x;  % Given trajectory for part a
nf = 3;  % Number of landmarks/features
T = 1;  % Sample time
I = eye(2);
Z = zeros(2);

R = 10^2*eye(2);

%% a:
% The location of the sensor is given.  The location of the landmarks can
% be solved as a linear weighted least squares (WLS) problem.  It is of
% course okay to solve this without SigSys.
%
% The measurement model is
%   y^i_k = m^i-x_k + e_k,
% stacked for i=1,2,3 for each time k.

mm = sensormod(@(t, x, u, th) th - repmat(x, [3, 1]), [2, 0, 2*nf, 2*nf]);
mm.pe = blkdiag(R, R, R);
Y = sig(ex2_y', 1, [], X');
M = calibrate(mm, Y);
figure(1); clf;
plot(M); hold on;
xplot2(Y)
% Fancy plotting solution (not needed), simply providing the covariance matrix is enough.
Mposa = reshape(M.th, [2, 3])
MPa = cat(3, M.P(1:2, 1:2), M.P(3:4, 3:4), M.P(5:6, 5:6))
for i=1:nf
  plot2(ndist(Mposa(:,i), MPa(:, :, i)))
end
print(1, '-depsc', fullfile('fig', 'ex2a'));

%% b:
% Extend to a complete SLAM problem, assume a 2D CV model for the sensor
% motion, and simply augment with the landmark positions.
F = [I, T*I; Z, I];
Ff = eye(2*nf);
Gf = zeros(2*nf, 2);  % How landmarks are affect by process noise.
G = [0.5*T^2*I; T*I];
% Tune the process noise to get reasonable performance, no huge maneuvers expected.
Q = diag([1 1].^2);
Q = [G; Gf]*Q*[G; Gf]';

% The measurement is relative pos (m^i) for each landmark.
```

```
H = repmat([-I Z], [nf, 1]);
Hf = eye(2*nf);  % How should landmarks enter the measurement
Rtot = blkdiag(R, R, R);

% Use a linear model to simplify.
modelb = lss(blkdiag(F, Ff), [], [H, Hf], [], Q, Rtot, 1);

% Create a sig object for the measurement, assume the landmark positions are input.
Yb = sig(ex2_y', 1);
% Initialize the filter in the middle of the landmarks, large uncertainy.
x0 = [0; 0; 0; 0];  P0 = diag([0, 0, 10, 10].^2);
xf0 = zeros(2*nf, 1);  Pf0 = kron(1e4^2*eye(2), eye(nf));

Xhatb = kalman(modelb, Yb, 'alg', 2, 'x0', [x0; xf0], 'P0', blkdiag(P0, Pf0));  % Filter

figure(2); clf;  % Produce the requested plots
xplot2(Xhatb, 'conf', 90); hold on

% Fancy plotting solution (not needed), simply providing the covariance matrix is enough.
Mposb = reshape(Xhatb.x(end, end-2*nf+1:end), [2, nf])
MPb = Xhatb.Px;
MPb = cat(3, squeeze(MPb(end, 5:6, 5:6)),...
             squeeze(MPb(end, 7:8, 7:8)),...
             squeeze(MPb(end, 9:10, 9:10)))
for i=1:nf
  plot2(ndist(Mposb(:,i), MPb(:, :, i)))
end
xlim([-300 100])

print(2, '-depsc', fullfile('fig', 'ex2b'));

%% c:
% Observations:
% * In (a) "global" positions are obtained whereas in (b) the assumed
%   initial position will remain undetermined.  The SLAM problem is not
%   observable with respect to the initial position.
% * The estimated landmark positions are much more uncertain in (b) than in
%   (a), as a result of the trajectory being uncertain.  The uncertainty
%   depends on the process noise, the less process noise, the more certain
%   landmark positions.
```

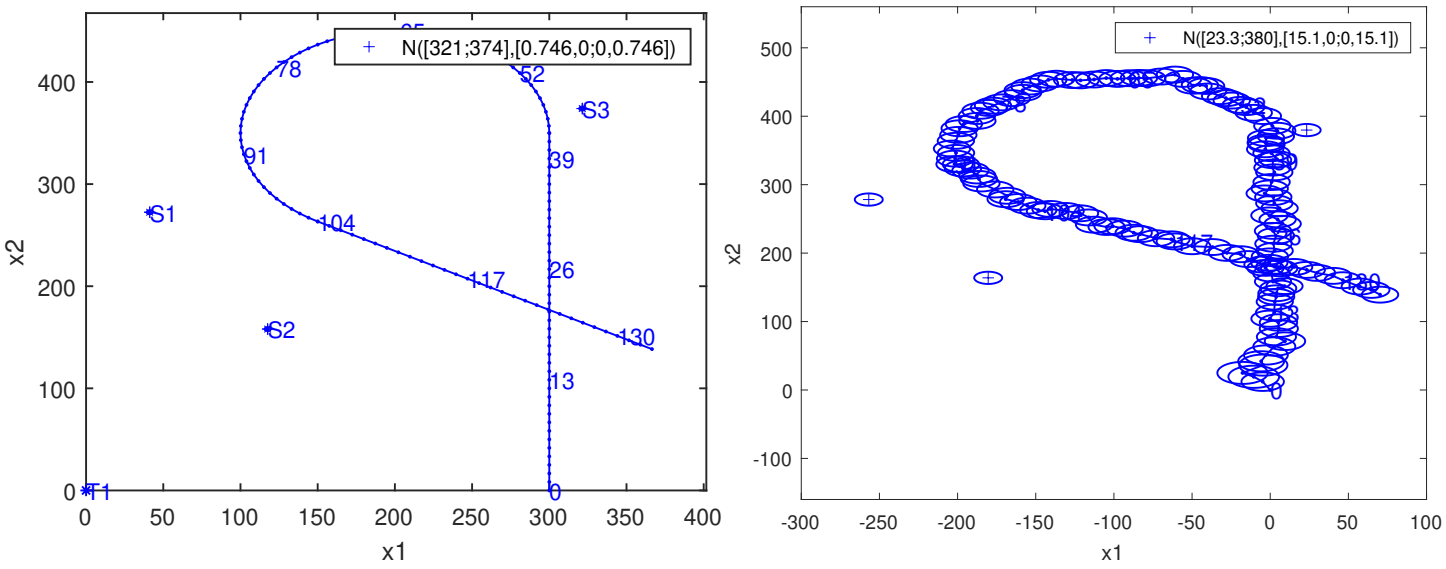

Figure 1: Resulting figures Exercise 2.

3. (a) For convenience, the Kalman filter recursion is given by the time update

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k} \tag{1a}$$

$$P_{k+1|k} = FP_{k|k}F^T + Q \tag{1b}$$

and measurement update

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + P_{k+1|k}H^T(HP_{k+1|k}H^T + R)^{-1}(y_k - H\hat{x}_{k+1|k}) \tag{1c}$$
$$P_{k+1|k+1} = P_{k+1|k} - P_{k+1|k}H^T(HP_{k+1|k}H^T + R)^{-1}HP_{k+1|k} \tag{1d}$$

Make the assumption that $\hat{x}^B_{k|k} = \hat{x}^A_{k|k}$ and $P^B_{k|k} = \gamma P^A_{k|k}$, and show by induction that this is in fact the case. The assumption gives

$$\hat{x}^B_{k+1|k} = F\hat{x}^B_{k|k} = F\hat{x}^A_{k|k} \Longrightarrow \hat{x}^B_{k+1|k} = \hat{x}^A_{k+1|k}$$
$$P^B_{k+1|k} = FP^B_{k|k}F^T + Q^B = F\gamma P^A_{k|k}F^T + \gamma Q^A = \gamma(P^A_{k|k}F^T + Q^A) = \gamma P^A_{k+1|k} \Longrightarrow P^B_{k+1|k} = \gamma P^A_{k+1|k},$$

that is, the time update preserves the relation between the estimates. Next, consider the measurement update given then

$$\begin{aligned}
\hat{x}^B_{k+1|k+1} &= \hat{x}^B_{k+1|k} + P^B_{k+1|k}H^T(HP^B_{k+1|k}H^T + R^B)^{-1}(y_k - H\hat{x}^B_{k+1|k}) \\
&= \hat{x}^A_{k+1|k} + \gamma P^A_{k+1|k}H^T(H\gamma P^A_{k+1|k}H^T + \gamma R^A)^{-1}(y_k - H\hat{x}^A_{k+1|k}) \\
&= \hat{x}^A_{k+1|k} + P^A_{k+1|k}H^T(HP^A_{k+1|k}H^T + R^A)^{-1}(y_k - H\hat{x}^A_{k+1|k}) = \hat{x}^A_{k+1|k+1} \Longrightarrow \hat{x}^B_{k+1|k+1} = \hat{x}^A_{k+1|k+1} \\
P^B_{k+1|k+1} &= P^B_{k+1|k} - P^B_{k+1|k}H^T(HP^B_{k+1|k}H^T + R^B)^{-1}HP^B_{k+1|k} \\
&= \gamma P^A_{k+1|k} - \gamma P^A_{k+1|k}H^T(H\gamma P^A_{k+1|k}H^T + \gamma R^A)^{-1}H\gamma P^A_{k+1|k} \\
&= \gamma P^A_{k+1|k} - \gamma P^A_{k+1|k}H^T(HP^A_{k+1|k}H^T + R^A)^{-1}HP^A_{k+1|k} \\
&= \gamma\Big(P^A_{k+1|k} - P^A_{k+1|k}H^T(HP^A_{k+1|k}H^T + R^A)^{-1}HP^A_{k+1|k}\Big) = \gamma P^A_{k+1|k+1} \Longrightarrow P^B_{k+1|k+1} = \gamma P^A_{k+1|k+1},
\end{aligned}$$

that is, the measurement update keeps the relationship, too.

To summarize, the exercise gives that $\hat{x}^B_{0|0} = \hat{x}^A_{0|0}$ and $P^B_{0|0} = \gamma P^A_{0|0}$. This serves as the starting point for induction. Next, the expressions just derived can then be used to propagate the property to $\hat{x}^B_{k|k} = \hat{x}^A_{k|k}$ and $P^B_{k|k} = \gamma P^A_{k|k}$ for $k \geq 0$.

(b) The same property as in the linear Kalman filter case does also apply for the regular EKF. Note that in this case $F = \nabla_x f(x)$ and $H = \nabla_x h(x)$ in each step, based on the previous estimate. However as the state estimates in the previous steps are all identical for the two filters, the linearization points used in the two filters are identical, and hence the linearizations too. With this additional argument the derivation in (a) still holds.

(c) The Kalman filter property does not hold for the UKF in general. This follows from the fact that when sigma points are selected, the covariance matrix P is used, yielding different sigma points with the two different tunings. Different sigma points yields different results when used in the algorithm, hence the result. (A notable exception is the linear case, where the property holds.)

4. (a) Spurious detection of photons not reflected in a surface is said to be uniformly distributed in the detection volume, hence

$$p(y|\mathcal{H}_0) = \mathcal{U}(y; 250, 350) = 1/100.$$

(b) When a surface is present at distance x, $P_d = 0.4$ of the detected photons have reflected in the surface, in which case they have a Gaussian error distribution, whereas the remaining ones are uniformly distributed in the detection volume, hence

$$\begin{aligned}
p(y|x, \mathcal{H}_1) &= P_d\mathcal{N}(y; x, R) + (1 - P_d)\mathcal{U}(y; 250, 350) \\
&= \frac{0.4}{\sqrt{2\pi \cdot 0.0043}} e^{-\frac{(y-x)^2}{2 \cdot 0.0043}} + 0.006.
\end{aligned}$$

(c)
```
%% Exercise 4
load data20230816

%% 4c
% Given information
Pd = 0.4;
R = 0.0043;
range = 350-250;

p1 = @(x, y) Pd*1/(sqrt(2*pi*R))*exp(-(x-y).^2/(2*R)) + (1-Pd)*1/range;
p0 = @(x, y) 1/range;

N = numel(ex4_y);
X = 250:0.01:350;  % Grid for plotting the LLR
LLR = zeros(size(X));
for i=1:numel(LLR)  % For simplicity, compute the LLR one x at the time
  LLR(i) = sum(log(p1(X(i), ex4_y)/p0(X(i), ex4_y)));
end

figure(400); clf;
plot(X, LLR); hold on;
plot(xlim, [0 0], '-.k');
xlabel('Distance [m]');
ylabel('LLR');

% Automate finding the peaks.  Each peak is to be considered a possible
% reflecting surface (cf GLR, where the ML estimate of x would be used).
% Only peaks where existance of surface (H1) is more likely than no surface
% (H0) are extracted.  Arguments can be made for setting this threshold
% lower and at least including one more surface.
[~, I] = findpeaks(LLR, 'MinPeakHeight', 0);
disp(X(I))
%  315.2700  316.6400

% Indicate the peaks in the plot.
for i=I
  plot(X(i)*[1 1], ylim, 'r');
end

print(400, '-depsc', fullfile('fig', 'ex4'));
```

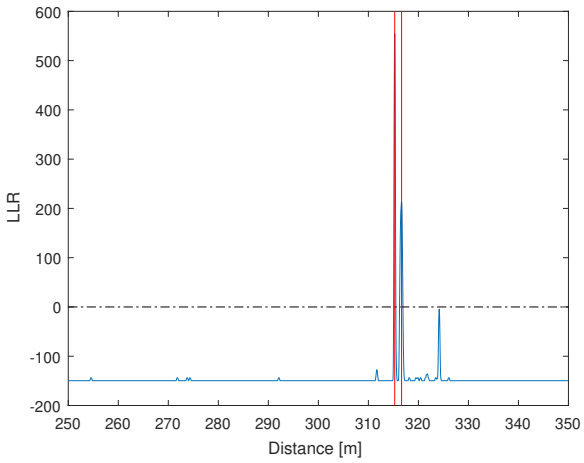

Figure 2: Resulting log-likelihood plot in Exercise 4.