Solutions for examination in Sensor Fusion, 2022-08-17

1. (a) **(iii), (iv), (v)**

   (b) $\boldsymbol{(H^T R^{-1} H)^{-1}}$

   For a linear Gaussian estimation problem as this, the weighted least squares gives the minimum variance unbiased estimate. The covariance matrix follows from this.

   (c) $\boldsymbol{R_1}$**–(b),** $\boldsymbol{R_2}$**–(c),** $\boldsymbol{R_3}$**–(a)**

   The "smaller" the $\boldsymbol{R}$, the more the estimate is affected by the measurements.

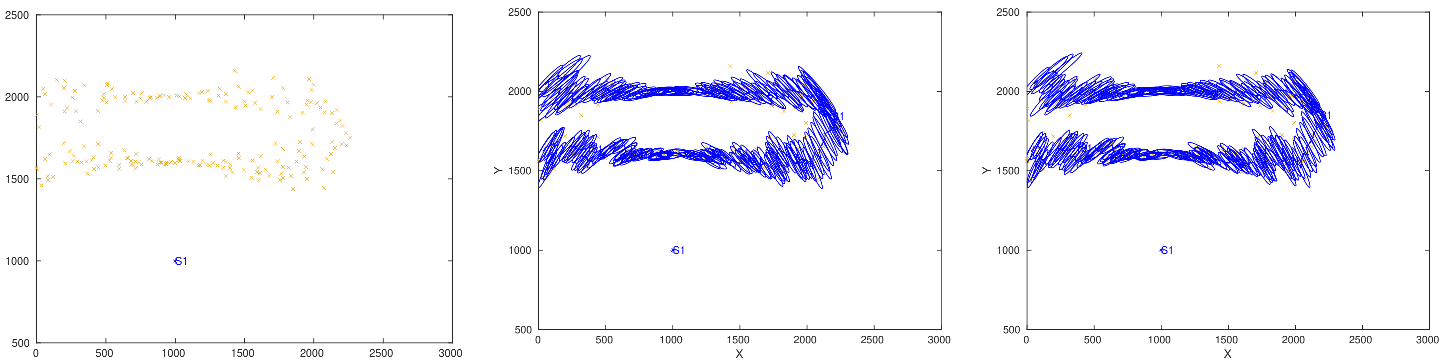
   (d) **(iii), (v)**

   (e) **Use an optimal proposal**

   The SNR is high, hence using an optimal is reasonable. Note, the structure of the problem does *not* lend itself to use the marginalized particle filter.

2.
```
%% Exercise 2
clear;
load data20220817
T = mean(diff(ex2_t));  % Sample time

%% Exercise 2a
sm = exsensor('radar');  % Make a radar measurement model
sm.th = [1000 1000]';  % Place the sensor at the correct location
R = diag([10, 0.1].^2);  % Set the measurement noise as given in the exercise
sm.pe = R;

Y = sig(ex2_y', ex2_t);  % Create sig object for the measuremnts

% Convert measurements to Cartesian coordinates
xcart = [ex2_y(1, :).*cos(ex2_y(2, :)); ex2_y(1, :).*sin(ex2_y(2, :))]+sm.th;

%Plot the result
figure(1); clf
plot(sm); hold on;
plot(xcart(1, :), xcart(2, :), 'x');
axis([0 3000, 500 2500])

% Given the plotted measurements, it seems the object moves mostly straight
% but makes a few turns.  A constant velocity model should capture this
% fairly ok, given enough process noise.
mm = exmotion('cv2d', T);
% Set the process noise, this is a tuning parameter
G = kron([T^2/2; T], eye(2));
q = 10.^2*eye(2);
mm.pv = G*q*G';


%% Exercise 2b
% Use the first two measurements for initialization
x0 = xcart(:, 1);  x1 = xcart(:, 2);
mm.x0 = [x0', (x1-x0)'/T];
mm.px0 = diag([100, 100, 100, 100].^2);  % Use reasonable inital uncertainty

mms = addsensor(mm, sm);  % Combined model
Xhat_ukf = ukf(mms, Y);  % Run filter
% Plot result
figure(2); clf;
plot(sm); hold on;
plot(xcart(1, :), xcart(2, :), 'x');
xplot2(Xhat_ukf, 'conf', 90);
axis([0 3000, 500 2500])

%% Exercise 2c

Xhat_pf = pf(mms, Y, 'Np', 1000);  % Run filter

figure(3); clf;
```

```
plot(sm); hold on;
plot(xcart(1, :), xcart(2, :), 'x');
xplot2(Xhat_pf, 'conf', 90);
axis([0 3000, 500 2500])

print(1, '-depsc', fullfile('fig', 'ex2a'))
print(2, '-depsc', fullfile('fig', 'ex2b'))
print(3, '-depsc', fullfile('fig', 'ex2c'))
```



(a) (b) (c)

Figure 1: Figures for Exercise 2.

3. (a) The likelihood for this problem is given by

$$p(y|x) = \tfrac{1}{2\pi} e^{-\frac{1}{2}\left((y_1-\cos(x))^2+(y_2-\sin(x))^2\right)}$$

The MLE is

$$\hat{x} = \arg\max_x p(y|x) = \arg\min_x -2\log\big(p(y|x)\big) = \arg\min_x\big((y_1-\cos(x))^2+(y_2-\sin(x))^2 - 2\log(2\pi)\big),$$

using that log is strictly increasing, and where the last constant does not affect the result and can be ignored.

To find an extreme point, differentiate and set to 0:

$$\tfrac{d}{dx}: \quad 0 = 2\big(y_1-\cos(x)\big)\sin(x) - 2\big(y_2-\sin(x)\big)\cos(x) = 2\big(y_1\sin(x) - y_2\cos(x)\big) \Leftrightarrow$$
$$\tan(x) = \tfrac{y_2}{y_1} \Rightarrow \hat{x} = \arctan(\tfrac{y_2}{y_1}).$$

The second derivative yields $y_1\cos(x) + y_2\sin(x)$ which is greater 0, and hence the extreme point is a minimum, if a quadrant compensated arctan is used.

(b) Gauss approximation formula for

$$h(x,e) = \hat{x} = \arctan(y_2/y_1) = \arctan\left(\frac{\sin(x)+e_2}{\cos(x)+e_1}\right)$$

yields

$$\begin{aligned}\operatorname{var}(\hat{x}) = h'_e(x,0)R(h'_e(x,0))^T &= \sigma^2\left(\left(\frac{-\sin(x)/\cos^2(x)}{1+(\sin(x)/\cos(x))^2}\right)^2 + \left(\frac{1/\cos(x)}{1+(\sin(x)/\cos(x))^2}\right)^2\right)\\ &= \sigma^2\frac{\sin^2(x)+\cos^2(x)}{(\sin^2(x)+\cos^2(x))^2} = \sigma^2\big(\cos^2(x)+\sin^2(x)\big) = \sigma^2.\end{aligned}$$

(c) The MSE is given by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x-\hat{x})^2$$

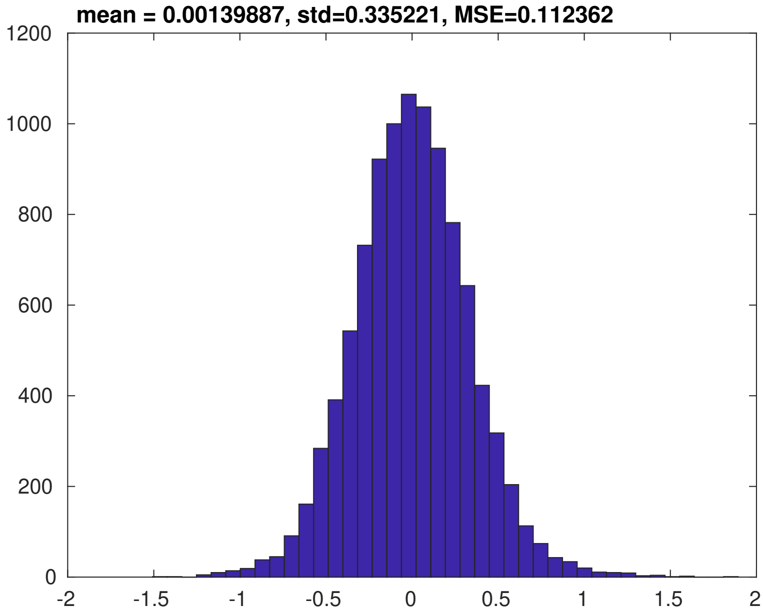


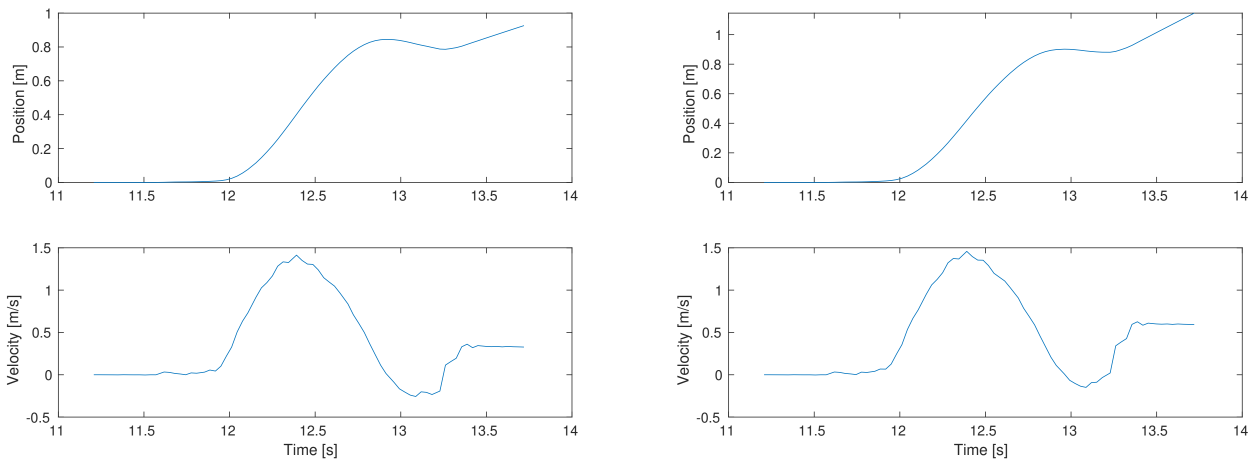
Figure 2: Results of Exercise 3c

```
N = 10000;
x0 = 0;
sigma = sqrt(0.1);
y = [cos(x0); sin(x0)] + sigma*randn(2, N);
xhat = atan2(y(2, :), y(1, :));
meanxhat = mean(xhat);
stdxhat = std(xhat);
msexhat = mean((xhat - meanxhat).^2);

figure(1); clf;
hist(xhat, 40);
title(sprintf('mean = %g, std=%g, MSE=%g', meanxhat, stdxhat, msexhat));
```

Results are shown in Figure 2.

4.
```
load('data20220817');

%% Ex 4a
% It is assumed the phone is horizontal all the time one, this can be
% used for calibration. The bias is the mean of measurements acquired
% during the calibration period (compensated with gravity g = 9.81).
%
% Assuming that the phone is moving only in the z-direction, the 1D dead
% reconing becomes a constant velocity model (using double integration).

% Indices for the two sequences:
cal_I = 200:330;
sub_I = 335:410;

% Calibrate the scenatio, by substracting mean acceleration compensated
% with for gravitaty, g:
g_acc = 9.81;
b = mean(ex4_y(:, cal_I), 2) - [0 0 g_acc]';
g = [0; 0; g_acc];
% Data for the time when the phone is in the lifted:
acc = ex4_y(:, sub_I);
t = ex4_t(sub_I);
K = size(acc, 2);  % Number of samples

xa = zeros(2, K+1);  % pos, vel
acc_z = acc(3, :) - b(3) - g(3);  % Bias compensated acc in z

% Perform dead reconing:
for k = 2:K
  T = t(k) - t(k-1);  % Sample time
  xa(:, k+1) = [1 T; 0 1]*xa(:, k) + [T^2/2; T] *acc_z(k);
end
```

```
figure(1); clc
subplot(2, 1, 1);  plot(t, xa(1, 2:end));
ylabel('Position [m]');
subplot(2, 1, 2);  plot(t, xa(2, 2:end));
xlabel('Time [s]');  ylabel('Velocity [m/s]');
print(1, '-depsc', fullfile('fig', 'ex4a'));
% The table is probably 84 cm high, judging from the level out of the
% hight estimate at 12.9 s.  The estimate then once again takes off,
% which  ndicates we still have some uncompensated sensor bias.  A reason
% could be that the phone is not completely plumb at table.
%
% We could also try to improve the estimate by using the fact that the
% phone should be stationary at beginning and end of the motion.
% However, this is not the approach taken here.

%% b)
% The same as in (a), with the acceleration the norm of the acceleration,
%  corrected for bias and gravity.

xb = zeros(2, K+1);  % pos, vel

% Perform dead reconing:
for k = 2:K
  % Compute acc_z for sample k
  % Assume that acc_z is equal to the norm of the acceleration:
  acc_z = norm(acc(:, k) - b) - g(3);
  T = t(k) - t(k-1);  % Sample time
  xb(:, k+1) = [1 T; 0 1]*xb(:, k) + [T^2/2; T]*acc_z;
end

figure(2);  clc
subplot(2, 1, 1);  plot(t, xb(1, 2:end));
ylabel('Position [m]');
subplot(2, 1, 2);  plot(t, xb(2, 2:end));
xlabel('Time [s]');  ylabel('Velocity [m/s]');
print(2, '-depsc', fullfile('fig', 'ex4b'));

% The table is probably 90 cm high, judging from the level out of the
% hight estimate at 13 s.  This is slightly higher than in (a), possibly
% because |y^a_k|=|a_k+e^a_k|$ has a positive bias, which in combination
% with the effect discussed in (a) could explain the faster increase in
% height at the end compared to (a).

%% c)
```



(a) (b)

Figure 3: Figures for Exercise 4(a) and 4(b).