Solutions for examination in Sensor Fusion, 2023-06-01

1. (a) **Put the IMUs flat on a level surface and collect data, and average to get $\bar{y}_a$, $\bar{y}_g$, and $\bar{y}_m$. For simplicity assume z facing up. Then $b_a = \bar{y}_a - g$ and $b_g = \bar{y}_g$. The magnetic field is much less predictable than the gravity, hence it's not possible to obtain a magnetometer bias estimate with this simple setup.**

   (b) **(i)–B, (ii)–C, (iii)–A, (iv)–D**

   (c) **$\hat{x} = 4.6$ and $P = 5.2$**

   From the exercise formulation, $\text{cov}(x_1, x_2) = 4$. Using this, the estimate is obtained using WLS.

   (d) **B and D**

   (e) **A *marginalized particle filtere* (MPF) is likely to improve the situation.**

   The models structure does have a conditional linear Gaussian substructure, conditioned on x^p. Hence, the MPF applies and is likely to improve the performance. As neither process noise nor measurement noise is extreme, the gain from a different proposal is probably limited.

2.
```
%% Ex 2
load('data20230601.mat');

M = ex2_m;  % Assumed landmark positions
nf = 3;  % Number of features
T = 1;  % Sample time
I = eye(2);
Z = zeros(2);

R = 10^2*eye(2);

%% a:
% Without any better information about the motion, assume it is well
% described by a CV model.  The state is position and velocity in 2D
Fa = [I, T*I; Z, I];
Gfa = zeros(4, 2*nf);  % How landmarks affect the state.
Ga = [0.5*T^2*I; T*I];
% Tune the process noise to get reasonable performance, no huge manuvers expected.
Q = diag([1 1].^2);
Qa = Ga*Q*Ga';

% The measurement is relative pos (m^i) for each landmark, for simplicity
% the landmarks are input to the model.
Ha = repmat([-I Z], [3, 1]);
Hfa = eye(2*nf);  % How should landmarks enter the measurement
Rtot = blkdiag(R, R, R);

% Use a linear model to simplify.
modela = lss(Fa, Gfa, Ha, Hfa, Qa, Rtot, 1);

% Create a sig object for the measurement, assume the landmark positions
% are input.
Ya = sig(ex2_y', 1, repmat(M(:)', [size(ex2_y, 2), 1]));
% Initialize the filter in the middle of the landmarks, large uncertainy.
x0a = [mean(M, 2); 0; 0];  P0a = diag([100, 100, 10, 10].^2);

Xhata = kalman(modela, Ya, 'alg', 2, 'x0', x0a, 'P0', P0a);  % Filter

figure(1); clf;  % Produce the requested plots
plot(M(1,:), M(2,:), 'xr'); hold on;
xplot2(Xhata, 'conf', 90);

print(1, '-depsc', fullfile('fig', 'ex2a'));

%% b:
% Uncertainty of landmarks
Rm = 50^2*eye(2);

% Augment the state with the landmarks, they are considered stationary.
% The matching model is then described as.
```

```
Fb = blkdiag(Fa, I, I, I);
Gb = [Ga; Z; Z; Z];  % No process noise on the landmarks
Gub = [];  % Ignore the input (now handled as state)
H = [Ha, Hfa];
Hfb = [];  % Ignore the input (now handled as state)
Qb = Gb*Q*Gb';

modelb = lss(Fb, Gub, H, Hfb, Qb, Rtot, 1);

% Create a sig object for the measurement, assume the landmark positions
% are input.
Yb = sig(ex2_y', 1);
% Initiate the motion part of the state as above, and the landmark part as
% described in the question.
x0b = [x0a; M(:)];
P0b = blkdiag(P0a, Rm, Rm, Rm);
Xhatb = kalman(modelb, Yb, 'alg', 2, 'x0', x0b, 'P0', P0b);  % Filter

figure(2); clf;  % Produce the requested plots
plot(M(1,:), M(2,:), 'xr'); hold on;
xplot2(Xhatb, 'conf', 90, 1:2); hold on;
xplot2(Xhatb(end), 'conf', 90, 'col', 'g', 5:6);
xplot2(Xhatb(end), 'conf', 90, 'col', 'm', 7:8);
xplot2(Xhatb(end), 'conf', 90, 'col', 'k', 9:10);

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

%% c:
% Rerun the estimation with "known" landmarks based on the estimates from b.

modelc = modela;  % Use the same model as in a, but with different input

% Some trickery is needed to extract the landmark postions...
Mest = Xhatb(end);
Mest = reshape(Mest.x(end-2*nf+1:end), [2, nf]);
Yc = sig(ex2_y', 1, repmat(Mest(:)', [size(ex2_y, 2), 1]));
x0c = x0a;
P0c = P0a;

Xhatc = kalman(modela, Yc, 'alg', 2, 'x0', x0a, 'P0', P0a);  % Filter

figure(3);  clf;  % Produce the requested plots
plot(M(1,:), M(2,:), 'xr'); hold on;
plot(Mest(1,:), Mest(2,:), 'mo');
xplot2(Xhata, Xhatc, 'conf', 90);

print(3, '-depsc', fullfile('fig', 'ex2c'));

% Observations:
% * The uncertainty in a) and c) are the same, the uncertainty for the
%   linear KF does not depend on the measurements.  If the model is the
%   same, the uncertainty will be the same.
% * The difference between the estimated trajectries is surprisingly small,
%   which is probably a consequence of the different errors in the map
%   pulling in different directions.  Had we had nonlinear measurements,
%   this would have been more troublesome.
% * The uncertainty in (b) is much higher, as it also has to compensate for
%   the uncertainty in the map.  Note, a large portion of the uncertainty
%   is shared, hence the relative uncertainty in map and position is
%   smaller.
```

3. (a) For a linear model, as the one in the exercise, the BLUE is given by the WLS estimator. The WLS estimate is given by, assuming independent identically distributed measurement noise,

$$\hat{x}^{\text{BLUE}} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{y} = \left(\sum_{k=1}^{N} H_k^T R^{-1} H_k\right)^{-1}\sum_{k=1}^{N} H_k^T R^{-1} y_k$$

$$P^{\text{BLUE}} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1} = \left(\sum_{k=1}^{N} H_k^T R^{-1} H_k\right)^{-1},$$

where $\mathsf{E}(e) = 0$ (due to symmetry), $R = \text{cov}(e_k)$, and bold quantities represent the stacked problem formulation. $R = 2\lambda^{-2}$, either look it up in a table (note, different parametrizations are used) or derive

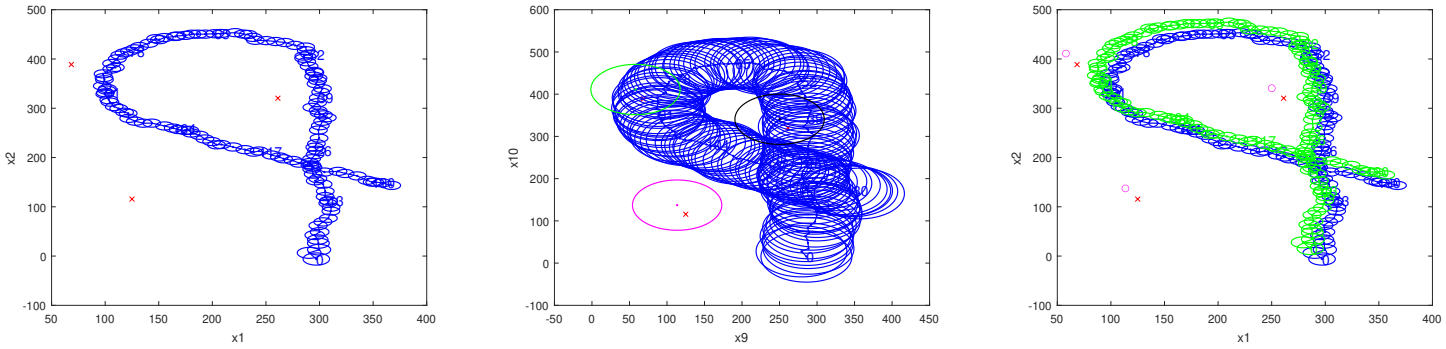


Figure 1: Resulting figures Exercise 2.

the result from the exponential distribution found in the textbook:

$$\begin{aligned}
\operatorname{cov}(e) &= \tfrac{1}{2}\lambda \int_{-\infty}^{+\infty} e^2 \exp(-\lambda|e|)\, de \\
&= \tfrac{1}{2}\lambda \int_{-\infty}^{0} e^2 \exp(-\lambda|e|)\, de + \tfrac{1}{2}\lambda \int_{0}^{+\infty} e^2 \exp(-\lambda|e|)\, de \\
&= \lambda \int_{0}^{+\infty} e^2 \exp(-\lambda e)\, de \\
&= \lambda \int_{0}^{+\infty} \left((e-\lambda)^2 + 2\lambda e - \lambda^2\right) \exp(-\lambda e)\, de \\
&= \lambda^{-2} + 2\lambda^{-2} - \lambda^{-2} = 2\lambda^{-2},
\end{aligned}$$

where the second to last step follow from mean and variance expressions for the exponential distribution. Hence:

$$\hat{x}^{\text{BLUE}} = \Big(\sum_{k=1}^{N} H_k^T H_k\Big)^{-1} \sum_{k=1}^{N} H_k^T y_k \qquad\qquad P^{\text{BLUE}} = 2\lambda^{-2}\Big(\sum_{k=1}^{N} H_k^T H_k\Big)^{-1},$$

(b) The measurements are independent (used in the first step), hence

$$\begin{aligned}
p(\mathbf{y}|x) &= \prod_{k=1}^{N} p(y_k|x) = \prod_{k=1}^{N} p(y_k - Hx) = \prod_{k=1}^{N} \tfrac{1}{2}\lambda \exp(-\lambda|y_k - Hx|) \\
&= 2^{-N}\lambda^N \exp(-\sum_{k=1}^{N} \lambda|y_k - Hx|)
\end{aligned}$$

Now the ML estimator follows as:

$$\begin{aligned}
\hat{x}^{\text{ML}} &= \arg\max_x p(\mathbf{y}|x) = \arg\min_x -\log(p(\mathbf{y}|x)) \\
&= \arg\min_x N\log 2 - N\log\lambda + \sum_{k=1}^{N} \lambda|y_k - Hx| \\
&= \arg\min_x \sum_{k=1}^{N} |y_k - Hx|,
\end{aligned}$$

where in the last step a constant contribution and a positive factor are removed to simplify the expression. The result shows that the ML estimate minimize the sum of absolute values of the difference between measurement and predicted measurement.

(c) To derive the CRLB, first compute

$$\begin{aligned}\log(p(y_k|x)) &= -\log 2 + \log\lambda - \lambda|y_k - H_k x| \\ \nabla_x \log(p(y_k|x)) &= \lambda H_k^T \operatorname{sgn}(y_k - H_k x) \\ \big(\nabla_x \log(p(y_k|x))\big)\big(\nabla_x \log(p(y_k|x))\big)^T &= \lambda^2 H_k^T H_k,\end{aligned}$$

where the derivative of the absolute value can be split in two parts depending on the sign of the argument.

The Fisher information now follows directly,

$$\begin{aligned}\mathcal{I}_k &= \int_{-\infty}^{\infty} \big(\nabla_x \log(p(y_k|x))\big)\big(\nabla_x \log(p(y_k|x))\big)^T dy_k = \int_{-\infty}^{\infty} \lambda^2 H_k^T H_k p(y_k|x)\, dy_k \\ &= \lambda^2 H_k^T H_k, \\ I &= \sum_{k=1}^{N} \mathcal{I}_k = \lambda^2 \sum_{k=1}^{N} H_k^T H_k.\end{aligned}$$

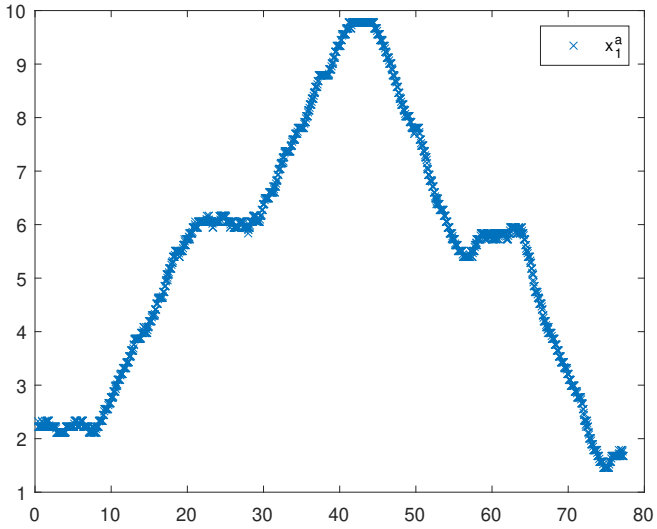
It is also just fine to derive the Fisher information wrt $H_k x$ for the parts, and then use (C.2) to get the Fisher information wrt x.

The CRLB can now be computed

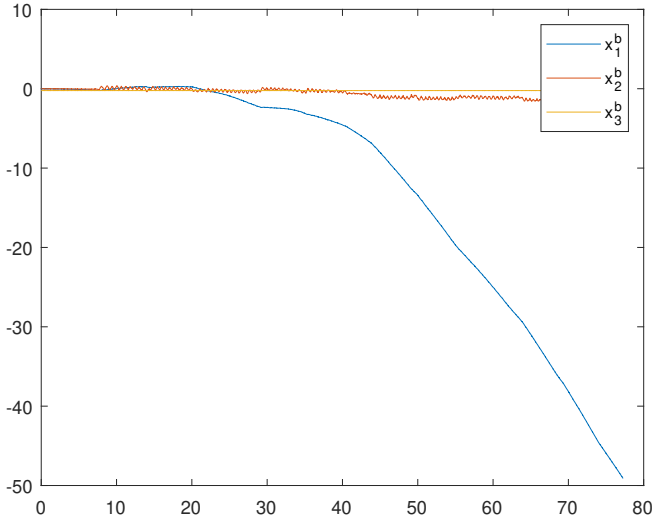
$$P^{\text{CRLB}} = \mathcal{I}^{-1} = \lambda^{-2}\Big(\sum_{k=1}^{N} H_k^T H_k\Big)^{-1}.$$

Note, the CRLB is half the covariance of the BLUE estimate covariance, hence there is room for improvement using a nonlinear estimator.

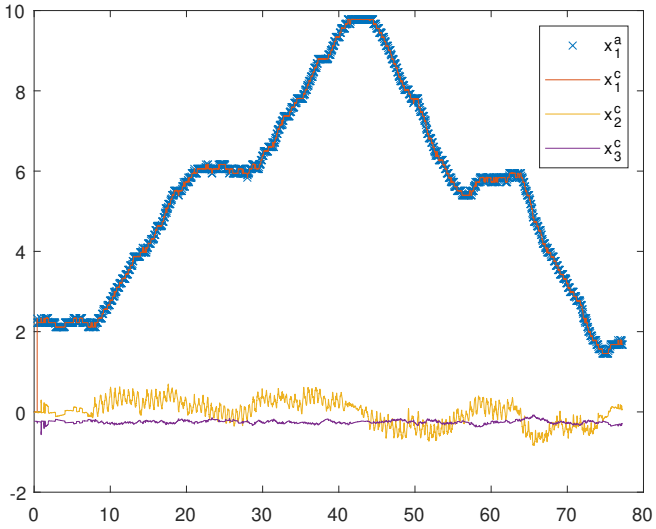
4.
```
load data20230601
%% Ex 4a
% Calibrate the barometer model: y = C_1*h+ C_0 + e
prs_0cm = mean(ex4_c0cm_prs);
prs_78cm = mean(ex4_c78cm_prs);
C0 = prs_0cm;
C1 = (prs_78cm - prs_0cm) / 0.78;
% Estimate the measurement noise, cov(e) = R
R = var([ex4_c0cm_prs - prs_0cm; ex4_c78cm_prs - prs_78cm]);

hhat = (ex4_prs - C0)/C1;  % Compute point estimates
figure(1); clf;  % Plot the point estimates
plot(ex4_t, hhat, 'x'); legend('x^a_1');

%% Ex 4b, c
% Assume a constant velocity height model extended with acc bias:
% x = [h; v; acc_bias];
% h_k+1 = h_k + T*v + T^2/2*(acc_y-g-bias) + T^2/2*w^v
% v_k+1 = v_k + T*(acc_y-g-bias) + T*w^v
% biax_k+1 = bias_k + w^b_k
g = 9.82;  % Gravity
q = 0.1^2;  % Process noise, ie accelerometer noise.
% Estimate the accelerometer bias based on the calibration sequences
bias = mean([ex4_c0cm_acc;  ex4_c78cm_acc]); bias = bias(3) - g;

xsim = [0; 0; bias];  % Results for 4b, x0
xhat = [0; 0; bias];  % Results for 4c, x0
P = diag([2, .1, 1].^2);  % P0
H = [C1 0 0];
for k=2:numel(ex4_t)  % Iterate over measurements
  T = ex4_t(k) - ex4_t(k-1);  % Time step
  u = ex4_acc(k-1, 3) - g;  % Input, based on accelerometer
  F = [1, T, -T^2/2;  0, 1, -T;  0, 0, 1];   % F(T)
  G = [T^2/2; T; 0];  % G(T)
  Q = G*q*G';  Q(3, 3) = T*.01;  % Q(T);
  % Time update
  xsim(:, k) = F*xsim(:, k-1) + G*u;  % 4b
  xhat(:, k) = F*xhat(:, k-1) + G*u;  % 4c
```



(a) Exercise 4(a).



(b) Exercise 4(b).



(c) Exercise 4(a, c)

Figure 2: Plots for exercise 4.

```
      P(:, :, k) = F*P(:, :, k-1)*F' + Q;
      % Measurement update
      if ~isnan(ex4_prs(k))
        yk = ex4_prs(k);  % Measurement in meter
        S = H*P(:,:, k)*H';
        K = squeeze(P(:, :, k))*H'/S;
        xhat(:, k) = xhat(:, k) + K*(yk-H*xhat(:, k) - C0);
        P(:, :, k) = (eye(3) - K*H)*squeeze(P(:, :, k))*(eye(3) - K*H)' + K*R*K';
      end
    end
    figure(2); clf;  % Plot results Ex 4b
    plot(ex4_t, xsim');  legend('x^b_1', 'x^b_2', 'x^b_3');
    figure(3); clf;  % Plot results Ex 4c
    plot(ex4_t, hhat, 'x');  hold on;
    plot(ex4_t, xhat');  legend('x^a_1', 'x^c_1', 'x^c_2', 'x^c_3');

    print(1, '-depsc', fullfile('fig', 'ex4a'));
    print(2, '-depsc', fullfile('fig', 'ex4b'));
    print(3, '-depsc', fullfile('fig', 'ex4c'));
```