SSY191 - Sensor Fusion and Nonlinear Filtering
Implementation of Home Assignment 02

Lucas Rath

Listings


```
clearvars; close all; clc;

%% 1) A)

close all;

% Motion Parameters
A = 1;
Q = 1.5;
% Measurement Parameters
H = 1;
R = 2.5;
% Prior
x_0 = 2;
P_0 = 6;


% General parameters
N = 20;
% calculate state and measurement sequences
X = genLinearStateSequence(x_0,P_0,A,Q,N);
Y = genLinearMeasurementSequence(X, H, R);


clr = fp.getColor(1:6);

figure('Color','white','Position',[586  360  505  359]);
hold on, grid on;
plot(0:N,X, 'b', 'DisplayName','true state sequence');
plot(1:N,Y,'*r', 'DisplayName','measurement sequence');
xlabel 'k - time step', ylabel 'value'
legend
% fp.savefig('q1a')


%% 1) B)

close all;

% filter data
% [x_k_k, P_k_k] = kalmanFilter(Y, x_0, P_0, A, Q, H, R);
[x_k_k, P_k_k, x_k_km1, P_k_km1, v, S, K] = kalmanFilter2(Y, x_0, P_0, A, Q, H, R);

% plot results
figure('Color','white','Position',[575  274  560  420]);
```

```
hold on, grid on;
p3 = plot([0:N], X, 'b', 'LineWidth',3, 'DisplayName','true state');
p3.Color = [p3.Color 0.2];
p2 = plot([0:N], [x_0 x_k_k], 'b', 'LineWidth',1.5, 'DisplayName','state estimate');
p1 = plot([1:N], Y, '*r', 'DisplayName','measurements');
p4 = plot([0:N], [x_0 x_k_k] + 3*sqrt([P_0 P_k_k(:)']), '--b', 'DisplayName','+3-sigma level');
p5 = plot([0:N], [x_0 x_k_k] - 3*sqrt([P_0 P_k_k(:)']), '--b', 'DisplayName','-3-sigma level');
xlabel('k - time step');
ylabel('x');
legend([p1 p2 p3 p4 p5],'Location','northeast');
% fp.savefig('q1b')


idx = [1 5 10 15];
for i=1:numel(idx)

    figure('Color','white','Position',[570  411  381  278]);
    hold on, grid on;

    xmu  = x_k_k(idx(i));
    xvar = P_k_k(idx(i));
    [x,y] = normpdf2(xmu, xvar, 4, 100);
    name = sprintf('p(x_{%d}|y_{1:%d})',idx(i),idx(i));
    p1 = plot(x,y, 'LineWidth',2, 'DisplayName', name);

    xtrue = X(idx(i)+1);
    p2 = plot([xtrue,xtrue], [0,max(y)*1.1],'--', 'LineWidth',3, 'Color', [clr(2,:) 0.5],'DisplayNam

    ylim([0,max(y)*1.1])
    title(['posterior distribution: $' name,'$'], 'Interpreter','Latex');
    ylabel(name), xlabel 'state x';
    legend('Location','southeast')
%     fp.savefig(sprintf('q1b_%d',idx(i)))
end


%% 1) C)

close all;

idx = [10, 15, 20];
for i=1:numel(idx)

    figure('Color','white','Position',[398  468  772  283]);
    hold on, grid on;

    mu_prior = x_k_k(idx(i)-1);
    P_prior  = P_k_k(idx(i)-1);
    [x,y] = normpdf2(mu_prior, P_prior, 4, 100);
    p1 = plot(x,y, 'LineWidth',2, 'DisplayName', sprintf('prior p(x_{%d}|y_{1:%d})',idx(i)-1,idx(i)-

    mu_pred = x_k_km1(idx(i));
    P_pred  = P_k_km1(idx(i));
    [x,y] = normpdf2(mu_pred, P_pred, 4, 100);
    p2 = plot(x,y, 'LineWidth',2, 'DisplayName', sprintf('predicted p(x_{%d}|y_{1:%d})',idx(i),idx(i

    mu_post = x_k_k(idx(i));
    P_post  = P_k_k(idx(i));
    [x,y] = normpdf2(mu_post, P_post, 4, 100);
    p3 = plot(x,y, 'LineWidth',2, 'DisplayName', sprintf('posterior p(x_{%d}|y_{1:%d})',idx(i),idx(i

    xtrue = X(idx(i)+1);
    p4 = plot([xtrue,xtrue], [0,max(y)*1.1],'--', 'LineWidth',3, 'Color', [clr(4,:) 0.5],'DisplayNam
```

```
    xmean = Y(idx(i));
    p4 = plot([xmean,xmean], [0,max(y)*1.1],'--', 'LineWidth',3, 'Color', [clr(5,:) 0.5],'DisplayNam

    ylim([0,max(y)*1.1])
    xlabel 'state x'
    legend('Location','southeast')
%     fp.savefig(sprintf('q1c_%d',idx(i)))
end


%% 1) D)
close all;

% General parameters
N = 1000;
% calculate state and measurement sequences
X = genLinearStateSequence(x_0,P_0,A,Q,N);
Y = genLinearMeasurementSequence(X, H, R);
% filter data
[x_k_k, P_k_k, x_k_km1, P_k_km1, v, S, K] = kalmanFilter2(Y, x_0, P_0, A, Q, H, R);


mean(x_k_k)
mean(v)

figure('Color','white','Position',[704  497  589  276]);
hold on, grid on;

histogram( (x_k_k-X(:,2:end)), 30 ,'DisplayName','histogram e=($x_k-\hat{x}_{k|k}$) ','Normalization

[x,y] = normpdf2(0, P_k_k(:,:,end), 4, 100);
p3 = plot(x,y, 'LineWidth',2, 'DisplayName', sprintf('gaussian N(x; 0, $P_{N|N})$') );

xlabel('e=($x_k-\hat{x}_{k|k}$)','Interpreter','Latex')
ylabel('normalized frequency','Interpreter','Latex')
title 'Histogram of normalized estimation error, N=1000'
legend('Interpreter','Latex')
% fp.savefig(sprintf('q1d'))


figure('Color','white','Position',[704  497  589  276]);
hold on, grid on;
autocorr(v)
xlabel 'Lag', ylabel 'Sample autocorrelation', title 'Sample autocorrelation function of innovation'
% fp.savefig(sprintf('q1d-aut'))



%% 1) E)

close all;

% General parameters
N = 20;
% calculate state and measurement sequences
X = genLinearStateSequence(x_0,P_0,A,Q,N);
Y = genLinearMeasurementSequence(X, H, R);

% Kalman filter with right x_0 and P_0
[x_k_k, P_k_k, x_k_km1, P_k_km1, v, S, K] = kalmanFilter2(Y, x_0, P_0, A, Q, H, R);
% Kalman filter with wrong x_0 and P_0
x_0w = 10;
P_0w = 6;
[x_k_kw, P_k_kw, x_k_km1w, P_k_km1w, vw, Sw, Kw] = kalmanFilter2(Y, x_0w, P_0w, A, Q, H, R);
```

```
% plot results
figure('Color','white','Position',[565  353  656  331]);
hold on, grid on;

p1 = plot([0:N], [x_0  x_k_k], 'Color',[clr(1,:),0.5], 'LineWidth',3, 'DisplayName','state estimate')
p2 = plot([0:N], [x_0  x_k_k] + 3*sqrt([P_0  P_k_k(:)']), '--', 'Color',clr(1,:), 'LineWidth',2, 'Disp
p3 = plot([0:N], [x_0  x_k_k] - 3*sqrt([P_0  P_k_k(:)']), '--', 'Color',clr(1,:), 'LineWidth',2, 'Disp

p4 = plot([0:N], [x_0w  x_k_kw], 'Color',[clr(3,:),0.5], 'LineWidth',3, 'DisplayName','state estimate
p5 = plot([0:N], [x_0w  x_k_kw] + 3*sqrt([P_0w  P_k_kw(:)']), '--', 'Color',clr(3,:), 'LineWidth',2, '
p6 = plot([0:N], [x_0w  x_k_kw] - 3*sqrt([P_0w  P_k_kw(:)']), '--', 'Color',clr(3,:), 'LineWidth',2, '

p8 = plot([0:N], X, '-', 'Color',[0 0 0 1], 'LineWidth',1, 'DisplayName','true state');

p7 = plot([1:N], Y, '*r', 'DisplayName','measurements');

xlabel('k - time step');
ylabel('x');
legend([p1 p2 p3 p4 p5 p6 p8 p7],'Location','southeast');
% fp.savefig('q1e')



%% 2) A)

close all;

T = 0.01;

% Motion Parameters
A= [1 T; 0 1];
tao = [0;1];
Q = tao * 1.5 * tao';
% Measurement Parameters
H = [1 0];
R = 2;
% Prior
x_0 = [1;3];
P_0 = 4*eye(2);

% General parameters
N = 50;

% calculate state and measurement sequences
X = genLinearStateSequence(x_0,P_0,A,Q,N);
Y = genLinearMeasurementSequence(X, H, R);


% plot position
figure('Color','white','Position',[340  124  427  294]);
hold on, grid on;
plot(0:N,H*X, 'b', 'DisplayName','true state sequence');
plot(1:N,Y,'*r', 'DisplayName','measurement sequence');
xlabel 'k - time step', ylabel 'position'
legend('Location','northwest')
% fp.savefig('q2a-pos')


% plot velocity
figure('Color','white','Position',[781  127  447  294]);
hold on, grid on;
plot(0:N,X(2,:), 'b', 'DisplayName','velocity sequence');
```

```
xlabel 'k - time step', ylabel 'velocity'
legend('Location','northwest')
% fp.savefig('q2a-vel')


%% 2) B)

close all;

% filter data
[x_k_k, P_k_k, x_k_km1, P_k_km1, v, S, K] = kalmanFilter2(Y, x_0, P_0, A, Q, H, R);

% plot position
figure('Color','white','Position',[321  517  457  294]);
hold on, grid on;
p3 = plot(0:N, H*X, 'b', 'LineWidth',3, 'DisplayName','true state');
p3.Color = [p3.Color 0.2];
p2 = plot(0:N, H*[x_0 x_k_k], 'b', 'LineWidth',1.5, 'DisplayName','state estimate');
p1 = plot(1:N, Y, '*r', 'DisplayName','measurements');
p4 = plot(0:N, H*[x_0 x_k_k] + 3*sqrt([P_0(1) squeeze(P_k_k(1,1,:))']), '--b', 'DisplayName','+3-sig
p5 = plot(0:N, H*[x_0 x_k_k] - 3*sqrt([P_0(1) squeeze(P_k_k(1,1,:))']), '--b', 'DisplayName','-3-sig
xlabel('k - time step');
ylabel('position');
legend([p1 p2 p3 p4 p5],'Location','northwest'); ylim([-3 11])
% fp.savefig('q2b-pos')


Hv = [0 1];
% plot velocity
figure('Color','white','Position',[788  517  447  294]);
hold on, grid on;
p3 = plot(0:N, Hv*X, 'b', 'LineWidth',3, 'DisplayName','true state');
p3.Color = [p3.Color 0.2];
p2 = plot(0:N, Hv*[x_0 x_k_k], 'b', 'LineWidth',1.5, 'DisplayName','state estimate');
p4 = plot(0:N, Hv*[x_0 x_k_k] + 3*sqrt([P_0(2,2) squeeze(P_k_k(2,2,:))']), '--b', 'DisplayName','+3-
p5 = plot(0:N, Hv*[x_0 x_k_k] - 3*sqrt([P_0(2,2) squeeze(P_k_k(2,2,:))']), '--b', 'DisplayName','-3-
xlabel('k - time step');
ylabel('velocity');
legend([p2 p3 p4 p5],'Location','northwest'); ylim([-15 35])
% fp.savefig('q2b-vel')



%% 2) C)

close all;

Qi = [0.1 1 10 1.5];

% General parameters
N = 50;
% calculate state and measurement sequences
X = genLinearStateSequence(x_0,P_0,A,Q,N);
Y = genLinearMeasurementSequence(X, H, R);


for i=1:numel(Qi)
    [x_k_k, P_k_k, x_k_km1, P_k_km1, v, S, K] = kalmanFilter2(Y, x_0, P_0, A, tao*Qi(i)*tao', H, R);

    % plot position
    figure('Color','white','Position',[192  538  421  303]);
```

```
    hold on, grid on;
    p3 = plot(0:N, H*X, 'b', 'LineWidth',3, 'DisplayName','true state');
    p3.Color = [p3.Color 0.2];
    p2 = plot(0:N, H*[x_0 x_k_k], 'b', 'LineWidth',1.5, 'DisplayName','state estimate');
    p1 = plot(1:N, Y, '*r', 'DisplayName','measurements');
    p4 = plot(0:N, H*[x_0 x_k_k] + 3*sqrt([P_0(1) squeeze(P_k_k(1,1,:))']), '--b', 'DisplayName','+3
    p5 = plot(0:N, H*[x_0 x_k_k] - 3*sqrt([P_0(1) squeeze(P_k_k(1,1,:))']), '--b', 'DisplayName','-3
    xlabel('k - time step');
    ylabel('position');
    title(sprintf('Kalman filter, Q=%.1f',Qi(i)))
    legend([p1 p2 p3 p4 p5],'Location','southeast');
%     fp.savefig(sprintf('q2c-pos-Qi-%d',i));
end



%% Help functions

function [x,y] = normpdf2(mu, sigma2, level, N)
    x = linspace(mu-level*sqrt(sigma2), mu+level*sqrt(sigma2), N);
    y = normpdf(x, mu, sqrt(sigma2));
end
```