SSY191 - Sensor Fusion and Nonlinear Filtering
Implementation of Home Assignment 04

Lucas Rath

Listings


```
%% 1A
clear all; close all; clc;
cp = fp.getColor(1:10);

sigma_v = 1*1e-4;
sigma_w = pi/180 ;

% True track
% Sampling period
T = 0.1;
% Length of time sequence
K = 600;
% Allocate memory
omega = zeros(1,K+1);
% Turn rate
omega(200:400) = -pi/201/T;
% Initial state
x0 = [0 0 20 0 omega(1)]';
% Allocate memory
X = zeros(length(x0),K+1);
X(:,1) = x0;
% Create true track
for i=2:K+1
    % Simulate
    X(:,i) = coordinatedTurnMotion(X(:,i-1), T);
    % Set turn rate
    X(5,i) = omega(i);
end


% Prior information
x_0 = [0 0 0 0 0]';
P_0 = diag([10 10 10 5*pi/180 pi/180].^2);
% Sensor positions
sk = [280 -140]';


% measurement noise
R = diag([15 2*pi/180].^2);
% generate measurement sequence
h = @(x) rangeBearingMeasurements(x, sk);
Y = genNonLinearMeasurementSequence(X,h,R);


% Motion model
```

```
f = @(x) coordinatedTurnMotion(x,T);
Q = diag([0 0 T*sigma_v^2 0 T*sigma_w^2]);

% [xf, Pf, xp, Pp] = nonLinearKalmanFilter(Y, x_0, P_0, f, Q, h, R, 'CKF');
[xs, Ps, xf, Pf, xp, Pp] = nonLinRTSsmoother(Y, x_0, P_0, f, Q, h, R, @sigmaPoints, 'CKF');

% calcualte unfiltered position from sensors given angles
Xm = sk + Y(1,:).*[cos(Y(2,:));sin(Y(2,:))];


figure('Color','white','Position',[207    70  1355   528]);
subplot(1,2,1)
title('Filter')
plotTurnU( X, xf, Pf, Xm, sk, 'filter', 2)
subplot(1,2,2)
title('Smoother')
plotTurnU( X, xs, Ps, Xm, sk, 'smoother', 4)
% fp.savefig('q1a-pos')


figure('Color','white','Position',[428  692  930  207]);
plotTurnUError( T, xf, xs, X )
% fp.savefig('q1a-error')


%% 1B
close all;

Y(1,150) = norm(X(1:2,150)) + 100;


% [xf, Pf, xp, Pp] = nonLinearKalmanFilter(Y, x_0, P_0, f, Q, h, R, 'CKF');
[xs, Ps, xf, Pf, xp, Pp] = nonLinRTSsmoother(Y, x_0, P_0, f, Q, h, R, @sigmaPoints, 'CKF');

% calcualte unfiltered position from sensors given angles
Xm = sk + Y(1,:).*[cos(Y(2,:));sin(Y(2,:))];

figure('Color','white','Position',[207    70  1355   528]);
subplot(1,2,1)
title('Filter')
plotTurnU( X, xf, Pf, Xm, sk, 'filter', 2)
subplot(1,2,2)
title('Smoother')
plotTurnU( X, xs, Ps, Xm, sk, 'smoother', 4)
% fp.savefig('q1b-pos')


figure('Color','white','Position',[428  692  930  207]);
plotTurnUError( T, xf, xs, X )
% fp.savefig('q1b-error')


%% 2A
cp = fp.getColor(1:10);

close all;

T = 0.01;

% Motion Parameters
A = 1;
f = @(x) A*x;
Q = 1.5;
% Measurement Parameters
```

```
H = 1;
h = @(x) H*x;
R = 2.5;
% Prior
x_0 = 2;
P_0 = 6;

% number of time steps
N = 50;
% number of particles
Np = 100;
% resample particles?
% bResample = true;
% sigma for the approximation in plotPostPdf
sigma = 1;



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

% filter data
[xf, Pf] = kalmanFilter(Y, x_0, P_0, A, Q, H, R);
% [xs, Ps, xf, Pf, xp, Pp] = nonLinRTSsmoother(Y, x_0, P_0, f, Q, h, R, @sigmaPoints, 'CKF');



% figure('Color','white','Position',[292   475  1165   422]);
% subplot(1,2,1)
bResample = true;
plotFunc_handle_r  = @(k, Xk, Xkmin1, Wk, j) plotPostPdf(k, Xk, Wk, xf, Pf, bResample, sigma, gca);
% [xfpr, Pfpr, Xpr, Wpr] = pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, plotFunc_handle_r, []);
[xfpr, Pfpr, Xpr, Wpr] = pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, [], []);

% subplot(1,2,2)
bResample = false;
plotFunc_handle  = @(k, Xk, Xkmin1, Wk, j) plotPostPdf(k, Xk, Wk, xf, Pf, bResample, sigma, gca);
% [xfp, Pfp, Xp, Wp] = pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, plotFunc_handle, []);
[xfp, Pfp, Xp, Wp] = pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, [], []);

mse(X(2:end)-xf)
mse(X(2:end)-xfpr)
mse(X(2:end)-xfp)


% plot position
figure('Color','white','Position',[199   288  1152   457]);
hold on, grid on;
p1 = plot(0:N, H*X, 'b', 'LineWidth',4, 'DisplayName','true state');
p1.Color = [p1.Color 0.2];
p2 = plot(0:N, H*[x_0 xf],  'Color','k', 'LineWidth',2, 'DisplayName','KF estimate');
p_pfr = plot(0:N, H*[x_0 xfpr], 'Color',cp(5,:), 'LineWidth',2, 'DisplayName','PF estimate with resa
p_pf  = plot(0:N, H*[x_0 xfp],  'Color',cp(3,:), 'LineWidth',2, 'DisplayName','PF estimate without r
p4 = plot(1:N, Y, '*r', 'DisplayName','measurements');
% p5 = plot(0:N, H*[x_0 xf] + 3*sqrt([P_0(1) squeeze(Pf(1,1,:))']), '--b', 'DisplayName','+3-sigma l
% p6 = plot(0:N, H*[x_0 xf] - 3*sqrt([P_0(1) squeeze(Pf(1,1,:))']), '--b', 'DisplayName','-3-sigma l
xlabel('k - time step');
ylabel('position');
legend([p1 p2 p_pfr p_pf p4],'Location','southwest');
title('Comparison KF and PF')
% fp.savefig('q2a-pos')
```

```
[Xl, KF, pApprox_r] = plotFunc_handle_r (ts, Xpr(:,:,ts), Xpr(:,:,ts-1), Wpr(:,ts)', 0);
[Xl, KF, pApprox]   = plotFunc_handle   (ts, Xp(:,:,ts),  Xp(:,:,ts-1),  Wp(:,ts)',  0);

ts = 30; %19
fig = figure('Color','white','Position',[675  549  570  420]);
hold on, grid on;
p1 = plot( X([ts,ts]+1), [0,max([pApprox,pApprox_r])*1.2],'--' , 'Color','b','LineWidth', 4, 'Displa
p1.Color = [p1.Color 0.2];
set(fig, 'Name', ['p_',num2str(ts), '_', 'SIR']);
plot( Xl, pApprox,    'Color',cp(5,:),'LineWidth', 2, 'DisplayName','PF estimate with resampling')
plot( Xl, pApprox_r,  'Color',cp(3,:),'LineWidth', 2, 'DisplayName','PF estimate without resampling'
plot( Xl, KF, '-.',  'Color','k', 'LineWidth', 2, 'DisplayName','Kalman filter')
legend( 'Location', 'southeast')
title(['p(x_k |  y_{1:k}),   k=', num2str(ts)])
ylim([0,max([pApprox,pApprox_r])*1.2])
% fp.savefig(sprintf('q2a-dist-%d',ts))


%% 2B
close all;

Np = 50;

bAlpha = false;

ploth = @(k, Xk, Xkmin1, Wk, j) plotPartTrajs(k, Xk, Xkmin1, Wk, j, bAlpha);

figure('Color','white','Position',[292   475  1165   422]);

subplot(1,2,1)
hold on, grid on;
bResample = true;
pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, ploth,[]);
plot(0:length(X)-1, X, 'Color',cp(2,:),'LineWidth', 2)
xlabel 'time step', ylabel 'state value', title 'Particle trajectories with resampling'
ylim([-8,12])

subplot(1,2,2)
hold on, grid on;
bResample = false;
pfFilter(x_0, P_0, Y, f, Q, h, R, Np, bResample, ploth ,[]);
plot(0:length(X)-1, X, 'Color',cp(2,:),'LineWidth', 2)
xlabel 'time step', ylabel 'state value', title 'Particle trajectories without resampling'
ylim([-32,24])
% fp.savefig('q2b-w')
% fp.savefig('q2b-nw')


%% 3A

clear all; close all; clc;
cp = fp.getColor(1:10);


% Generate measurements
% [x,y] = MapProblemGetPoint(false);
slam = SLAM();
% slam.drawPath('Xk.mat')
slam.loadPosition('Xk.mat');

R = eye(2) * 0.01^2;
ddt = @(x,dt) conv2(x,[1 -1]/dt,'valid');
% measurements
```

```
v = ddt(slam.pos,1) + mvnrnd(zeros(2,1), R, length(slam.pos)-1)';



sigma_v = 2;%4;
sigma_w = deg2rad(20); %deg2rad(10);
T = 1;

% Motion model
f = @(x) coordinatedTurnMotion(x,T);
Q = diag([0 0 T*sigma_v^2 0 T*sigma_w^2]);

% Measurement model
h = @(x) [x(3,:).*cos(x(4,:));
          x(3,:).*sin(x(4,:))];
R = diag([0.2 0.2].^2);

N = 10000;
bResample = true;

% Prior distribution
x_0 = [slam.pos(:,1); 0.4; 0; 0];
P_0 = diag([0 0 0 deg2rad(180) deg2rad(5)].^2);

Xp0 = [ SLAM.genValidRandParticles(N);     % pos
        mvnrnd(x_0(3:5),P_0(3:5,3:5),N)'];

% [xfp, Pfp, Xp, Wp] = pfFilter(x_0,P_0,v,f,Q,h,R,N,bResample,[], @SLAM.isOnRoad);
[xfp, Pfp, Xp, Wp] = pfFilter(Xp0,P_0,v,f,Q,h,R,N,bResample,[], @SLAM.isOnRoad);


close all;
figure('Color','white','Position',[341   440   578   438]);
slam.plotmap()
p1 = plot( slam.pos(1,:)+slam.pos(2,:)*1i, '-*','Color',cp(2,:) , 'DisplayName','True trajectory')
legend(p1)
% fp.savefig('q3a')

%% plot SLAM
% close all;
% figure('Color','white','Position',[593    19  1249   835]);
figure('Color','white','Position',[1038    14   809   563]);
clf;
slam.plotmap()

pp = scatter( Xp0(1,:), Xp0(2,:) ,50, 'o', 'MarkerFaceAlpha',0.2, 'MarkerFaceColor', cp(4,:), 'Marke
ell_xy = sigmaEllipse2D(xfp(1:2,1),Pfp(1:2,1:2,1),3,50);
pe = fill(ell_xy(1,:),ell_xy(2,:), cp(5,:),'facealpha',.1);

% title(sprintf('Particle filter - Localization, k=%d',1))
% fp.savefig(sprintf('q3d-k-%d',1))

save_idx = [1 2 6 13 23 46 57 85 120 138];

pause(3)
K = length(xfp);
for k=2:K
    plot( xfp(1,k-1:k)+xfp(2,k-1:k)*1i, '-*','Color',cp(1,:) )
    plot( slam.pos(1,k-1:k)+slam.pos(2,k-1:k)*1i, '-*','Color',cp(2,:) )
    pp.XData = Xp(1,:,k);
    pp.YData = Xp(2,:,k);

    ell_xy = sigmaEllipse2D(xfp(1:2,k),Pfp(1:2,1:2,k),3,50);
    pe.XData = ell_xy(1,:);
```

```
    pe.YData = ell_xy(2,:);

    title(sprintf('Particle filter - Localization, k=%d',k))
    drawnow()
    pause(0.1)

    if any(k==save_idx)
%         fp.savefig(sprintf('q3c-k-%d',k))
%         fp.savefig(sprintf('q3d-k-%d',k))
    end
end


figure

subplot(1,3,1);hold on;
plot(v(1,:))
plot(xfp(3,:).*cos(xfp(4,:)))
subplot(1,3,2);hold on;
plot(v(2,:))
plot(xfp(3,:).*sin(xfp(4,:)))
subplot(1,3,3);hold on;
plot(vecnorm(v))
plot(xfp(3,:))

% plot( wrapTo180( rad2deg(xfp(4,:)) ))


%% help functions


function plotTurnU( X, xf, Pf, Xm, sk, signame, coln)
    cp = fp.getColor(1:10);
    grid on; hold on, axis equal;
    for i=1:5:length(xf)
        ell_xy = sigmaEllipse2D(xf(1:2,i),Pf(1:2,1:2,i),3,50);
        p5 = fill(ell_xy(1,:),ell_xy(2,:), cp(coln,:),'facealpha',.1, 'DisplayName',[signame,' 3-sig
%,'edgecolor','none'
    end

    p1 = plot(X(1,:),X(2,:),   '-', 'Color', cp(1,:), 'LineWidth',2, 'DisplayName','True position se
    p2 = plot(xf(1,:),xf(2,:), '-', 'Color', cp(coln,:), 'LineWidth',2, 'DisplayName',[signame,' pos
    sc1 = scatter(sk(1), sk(2), 100, 'o', 'MarkerFaceAlpha',0.8, 'MarkerFaceColor', cp(4,:), 'Marker

    axis manual
    p4 = plot(Xm(1,:),Xm(2,:), 'Color', [cp(3,:) 0.3], 'LineWidth',1, 'DisplayName','Measured positi

    xlabel 'pos x', ylabel 'pos y'
    legend([p1 p2 p4 sc1 p5], 'Location','west')
    % if savefig fp.savefig(sprintf('q3_%s',name2save)); end
end

function plotTurnUError( T, xf, xs, X )
    cp = fp.getColor(1:10);
    K = length(X)-1;
    grid on, hold on;
    p1 = plot( (1:K)*T, vecnorm(xf(1:2,:)-X(1:2,2:end), 2, 1), 'Color', cp(2,:) , 'LineWidth',2, 'Di
    p2 = plot( (1:K)*T, vecnorm(xs(1:2,:)-X(1:2,2:end), 2, 1), 'Color', cp(4,:) , 'LineWidth',2, 'Di
    ylabel('$|p_k - \hat{p_{k|k}}|_2$', 'Interpreter','Latex', 'FontSize',16), xlabel('Time [s]')
    title 'Position error'
    legend([p1 p2])
    % if savefig fp.savefig(sprintf('q3_%s_err',name2save)); end
end
```