```
%% Analysis code
clear all;
close all;
clc;

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(150:450) = -pi/301/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
S1 = [300 -100]';
S2 = [300 -300]';
S = [S1,S2];
% measurement noise
R = diag([pi/180 pi/180].^2);
% generate measurement sequence
h = @(x)dualBearingMeasurement(x,S1,S2);
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, T, Q, S, h, R,
@sigmaPoints, 'CKF');

% calcualte unfiltered position from sensors given angles
Xm = S1 + Y(1,:).*[cos(Y(2,:));sin(Y(2,:))];
figure
subplot(1,2,1)
title('Filter')
plotTurnU( X, xf, Pf, Xm, S1, 'filter', 2)
subplot(1,2,2)
```

```
title('Smoother')
plotTurnU( X, xs, Ps, Xm, S1, 'smoother', 2)
figure
plotTurnUError( T, xf, xs, X )

%% 1B
close all;

Y(:,300) = Y(:,300) + mvnrnd(zeros(1,2), 10*R)';
% [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, T, Q, S, h, R,
@sigmaPoints, 'CKF');

% calculate unfiltered position from sensors given angles
Xm = S1 + Y(1,:).*[cos(Y(2,:));sin(Y(2,:))];

figure
subplot(1,2,1)
title('Filter')
plotTurnU( X, xf, Pf, Xm, S1, 'filter', 2)
subplot(1,2,2)
title('Smoother')
plotTurnU( X, xs, Ps, Xm, S1, 'smoother', 4)
figure
plotTurnUError(T, xf, xs, X)


%% 2 - Analysis
clear all;
clc;
close all;
%% 2.2 a/b
clear;
clc;
close all;

rng(970427);

% Variances
Q = 1.5;
R = 3;

% Initial Prior
x0 = -20;
P0 = 2;

%Number of particles
N = 100;

% Parameters
A = 1; f = @(x) A*x; C = 1; h = @(x) C*x; T = 0.1; K = 3/T;

% Genarating Data State and Measurement Sequence
X = genLinearStateSequence(x0, P0, A, Q, K);
```

```
Y = genLinearMeasurementSequence(X, C, R);

%KALMANFILTER Filters measurements sequence Y using a Kalman filter.
[xf, xp] = kalmanFilter(Y, x0, P0, A, Q, C, R);

resampling = false; sigma = 1;
plotFunc_handle  = @(k, Xk, Xkmin1, Wk, j)(plotPostPdf(k, Xk, Wk, xf, xp,
resampling, sigma));

%Plot func not in arguments
[xfp, Pfp, Xp, Wp] = pfFilter(x0, P0, Y, f, Q, h, R, N, resampling,
plotFunc_handle);

resampling = true;
plotFunc_handle  = @(k, Xk, Xkmin1, Wk, j) plotPostPdf(k, Xk, Wk, xf, xp,
resampling, sigma);

%Plot func not in arguments
[xfp_resamp, Pfp_resamp, Xp_resamp, Wp_resamp] = pfFilter(x0, P0, Y, f, Q, h,
R, N, resampling, plotFunc_handle);

% plot for Compare the performance of the PF with a KF in terms of mean
square error (MSE)
figure(1);
clf;
hold on;
grid on;
% True state
plot((0:K) ,X, 'r', 'LineWidth', 2)
% Measurements
plot((1:K),Y, '*g')

% Kalman filter estimate
plot((0:K), [x0 xf], '-b', 'LineWidth', 2)

% Particle filter estimate without resampling
plot((0:K), [x0 xfp], '-k', 'LineWidth', 2)

% Particle filter estimate with resampling
plot((0:K), [x0 xfp_resamp], '-k', 'LineWidth', 2)

title('Kalman filter vs Particle filter')
xlabel('Time [s]'); ylabel('State value');
legend('True state ', 'Measurements', 'Kalman','PF without resampling','PF
with resampling','Interpreter','latex')

Kalman = mse(X(2:end)-xf)
pf1 = mse(X(2:end)-xfp)
pf2 = mse(X(2:end)-xfp_resamp)

time=1:K
B = reshape(xp,1,30)
error_kalman=errorbar(time,xf, sqrt(B),'LineWidth',1)
title('position vs time')
```

```
%% b/c
% K=30
figure(1);
clf;
subplot(2,1,1);
hold on;
grid on;
N = 100; %Number of particles
resampling = false;
rng(1997);
[xfp, Pfp, Xp, Wp] = pfFilter(x0, P0, Y, f, Q, h, R, N, resampling,
@plotPartTrajs); %Plot func not in arguments

plot(0:K, X, '-g', 'LineWidth', 1)    % True state
xlabel("Sample [k]")
ylabel ("State value")

subplot(2,1,2);
hold on;
grid on;
N = 100; %Number of particles
resampling = true;
rng(1997);
[xfp, Pfp, Xp, Wp] = pfFilter(x0, P0, Y, f, Q, h, R, N, resampling,
@plotPartTrajs); %Plot func not in arguments
plot(0:K, X, '-g', 'LineWidth', 1) % True state

xlabel("Sample [k]")
ylabel ("State value")

%% help functions
function plotTurnU( X, xf, Pf, Xm, S1, 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-sigma']);   %,'edgecolor','none'
    end

    p1 = plot(X(1,:),X(2,:),   '-', 'Color', cp(1,:), 'LineWidth',2,
'DisplayName','True position sequence');
    p2 = plot(xf(1,:),xf(2,:), '-', 'Color', cp(coln,:), 'LineWidth',2,
'DisplayName',[signame,' position']);
    sc1 = scatter(S1(1), S1(2), 100, 'o', 'MarkerFaceAlpha',0.8,
'MarkerFaceColor', cp(4,:), 'MarkerEdgeColor', cp(4,:),'DisplayName','sensor
1 location');

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

    xlabel 'pos x', ylabel 'pos y'
    legend([p1 p2 p4 sc1 p5], 'Location','best')
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, 'DisplayName','Filter error');
    p2 = plot( (1:K)*T, vecnorm(xs(1:2,:)-X(1:2,2:end), 2, 1), 'Color',
cp(4,:) , 'LineWidth',2, 'DisplayName','Smoother error');
    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
```