```
close all;
clear all;
clc;
% number of samples
N=10000;
% filter update types {'EKF', 'UKF', 'CKF'}
type = 'CKF';
% two prior distributions 1 or 2
distribution = '1';
switch distribution
case '1'
x_0mu = [120 120]';
P_0 = [25 0;0 100];
case '2'
x_0mu = [120 -20]';
P_0 = [25 0;0 100];
end
% positions of sensors
s_1 = [0; 100]';
s_2 = [100; 0]';
% Noise covariance
R = diag([0.1*pi/180 0.1*pi/180].^2);
% Measurement model from functions
h = @(x) dualBearingMeasurement(x,s_1,s_2);
[y_mu, y_sigma, y_s] = approxGaussianTransform(x_0mu, P_0,
 @(x)genNonLinearMeasurementSequence(x,h,R) , N);
% estimate transformed mean and covariance
switch type
case 'UKF'
[SP,W] = sigmaPoints(x_0mu,P_0,type);
hSP1 = h(SP);
[ye_mu, ye_sigma] = nonLinKFprediction(x_0mu, P_0, h, R, type);
case 'CKF'
[SP,W] = sigmaPoints(x_0mu,P_0,type);
hSP1 = h(SP);
[ye_mu, ye_sigma] = nonLinKFprediction(x_0mu, P_0, h, R, type);
case 'EKF'
[hx, dhx] = h(x_0mu);
[ye_mu, ye_sigma] = nonLinKFprediction(x_0mu, P_0, h, R, type);
end
% ploting ellipse level curve at 3sigma
level=3;
N2=200;
[ xy ] = sigmaEllipse2D( y_mu, y_sigma, level, N2 );
[ xye ] = sigmaEllipse2D( ye_mu, ye_sigma, level, N2 );
figure(1);
scatter(y_s(1,:), y_s(2,:), 10)
hold on
plot(xy(1,:), xy(2,:));
hold on
scatter(y_mu(1), y_mu(2), 10, 'o');
hold on
```

```
switch type
case 'UKF'
scatter(hSP1(1,:), hSP1(2,:));
case 'CKF'
scatter(hSP1(1,:), hSP1(2,:));
end
hold on
scatter(ye_mu(1,:), ye_mu(2,:))
hold on
plot(xye(1,:),xye(2,:))
xlabel 'y1- phi';
ylabel 'y2 - phi_2';
switch type
case 'UKF'
legend('Measurement distribution','Untransformed
 ellipse','untransformed mean','sigma points','transformed
 mean','transformed ellipse')
case 'CKF'
legend('Measurement distribution','Untransformed
 ellipse','untransformed mean','sigma points','transformed
 mean','transformed ellipse')
case 'EKF'
legend('Measurement distribution','Untransformed
 ellipse','untransformed mean','transformed mean','transformed
 ellipse')
end
```

Question 2 Non-linear Kalman Filtering

```
clc;
clear all;
close all;
x_0=[0 0 14 0 0]';
P_0= diag([100 100 4 (pi/180)^2 (5*pi/180)^2]);
 s_1=[-200,100]';
 s_2=[-200,-100]';
 T=1;
 N=100;
 Qtable='1';
 switch Qtable
        case '1'
            gamma=[ 0 0;0 0;1 0;0 0;0 1];
            Q=gamma*diag([1 pi/180].^2)*gamma';
            R=diag([10*pi/180  0.5*pi/180].^2);
        case '2'
            gamma=[ 0 0;0 0;1 0;0 0;0 1];
            Q=gamma*diag([1 pi/180].^2)*gamma';
            R=diag([0.5*pi/180  0.5*pi/180].^2);
 end

 motionModel = @(x) coordinatedTurnMotion(x, T);
 X = genNonLinearStateSequence(x_0, P_0, motionModel, Q, N);
 h = @(x) dualBearingMeasurement(x,s_1,s_2);
 Y = genNonLinearMeasurementSequence(X, h, R);


for type = {'EKF','UKF','CKF'}

    % filter
    [xf, Pf, xp, Pp] = nonLinearKalmanFilter(Y, x_0, P_0, motionModel,
 Q, h, R, type{1});
    xmeas = ( s_2(2)-s_1(2) + tan(Y(1,:))*s_1(1) -
 tan(Y(2,:))*s_2(1) ) ./ ( tan(Y(1,:)) - tan(Y(2,:)) );
    ymeas = s_1(2) + tan(Y(1,:)) .* ( xmeas(1,:) - s_1(1) );
    figure(1);
    grid on;
    hold on
    for i=1:5:length(xf)
        level=3;
        N2=100;
        Var_xy = sigmaEllipse2D(xf(1:2,i),Pf(1:2,1:2,i),level,N2);
        plot(Var_xy(1,:),Var_xy(2,:),'Color','black');
    end

    plot(X(1,:),X(2,:), 'Color','red');
    plot(xf(1,:),xf(2,:), 'Color', 'blue');

    scatter(s_1(1), s_1(2), 100, 'o');
    scatter(s_2(1), s_2(2), 200, 'h');

    axis manual
    plot(xmeas,ymeas,'*');

    xlabel 'pos x', ylabel 'pos y'
legend('variance','truepositions','filtered position','sensor1
 position','sensor2 position','Measurements')
end

MC = 100;

est_err = cell(1,3);

 type = {'EKF','UKF','CKF'};
for imc = 1:MC
    X = genNonLinearStateSequence(x_0, P_0, motionModel, Q, N);
    Y = genNonLinearMeasurementSequence(X, h, R);
     for itype = 1:numel(type)

        %  Kalman filter
        [xfM,PfM,xpM,PpM] =
 nonLinearKalmanFilter(Y,x_0,P_0,motionModel,Q,h,R,type{itype});
        est_err{1,itype}(1:2,end+1:end+length(xf)) = X(1:2,2:end) -
 xfM(1:2,:);

    end
end
```

```
 MCcount = 100;
 close all;
 loc = {'x','y'};
     figure(2);
     for itype = 1:numel(type)
        for iloc = 1:numel(loc)
            subplot(2,3, itype + (iloc-1)*numel(type) );
            hold on;
%
             histo = est_err{1,itype}(iloc,:);
             errmu  = mean(histo);
             errsig = std(histo);

            histogram( histo, MCcount ,'Normalization','pdf');
            level=3;
            N2=100;
            x = linspace(errmu-level*sqrt(errsig^2), errmu
+level*sqrt(errsig^2), N2);
            y = normpdf(x, errmu, sqrt(errsig^2));
            plot(x,y, 'LineWidth',2 );

        end
     end
```

part3

```
 clc
clear all;
close all;
% 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
x_0 = [0 0 0 0 0]';
P_0 = diag([10 10 10 5*pi/180 pi/180].^2);
```

```
% Sensor positions
s_1 = [280 -80]';
s_2 = [280 -200]';
gamma=[ 0 0;0 0;1 0;0 0;0 1];
Q=gamma*diag([0.1*1 0.1*pi/180].^2)*gamma';
% measurement variance
R = 0.05*diag([4*pi/180 4*pi/180].^2);
% generate measurement sequence
h = @(x) dualBearingMeasurement(x,s_1,s_2);
Y = genNonLinearMeasurementSequence(X,h,R);
% Motion model
motionModel = @(x) coordinatedTurnMotion(x,T);
[xf, Pf, xp, Pp] = nonLinearKalmanFilter(Y, x_0, P_0, motionModel, Q,
 h, R, 'CKF');
%unfiltered position
xmeas = ( s_2(2)-s_1(2) + tan(Y(1,:))*s_1(1) - tan(Y(2,:))*s_2(1) ) ./
( tan(Y(1,:)) - tan(Y(2,:)) );
ymeas = s_1(2) + tan(Y(1,:)) .* ( xmeas(1,:) - s_1(1) );
figure(1);
grid on;
hold on,
axis equal;
plot(X(1,:),X(2,:), 'Color','red');
plot(xf(1,:),xf(2,:), 'Color', 'blue');
scatter(s_1(1), s_1(2), 100, 'o');
scatter(s_2(1), s_2(2), 200, 'o');
axis manual
plot(xmeas,ymeas,'*');
for i=1:15:length(xf)
var_xy = sigmaEllipse2D(xf(1:2,i),Pf(1:2,1:2,i),3,50);
plot(var_xy(1,:),var_xy(2,:))
end
xlabel 'pos x';
ylabel 'pos y';
legend('true state','filtered position','sensor1 potion','sensor2
 potion','Measurements')
% plot position error
err=X(1:2,2:end) - xf(1:2,:);
figure(2);
grid on;
hold on;
plot( (1:K)*T,err(1,:),(1:K)*T,err(2,:) )


options = struct('format', 'pdf', 'evalCode',false);
publish('HA3_matlab_code.m', options);
publish('approxGaussianTransform.m', options);
publish('coordinatedTurnMotion.m', options);
publish('dualBearingMeasurement.m', options);
publish('genNonLinearMeasurementSequence.m', options);
publish('genNonLinearStateSequence.m', options);
publish('nonLinearKalmanFilter.m', options);

publish('nonLinKFprediction.m', options);
publish('nonLinKFupdate.m', options);
publish('sigmaEllipse2D.m', options);
publish('nonLinearKalmanFilter.m', options);
```

Published with MATLAB® R2020a