Assignment No: 5

Table of Contents

Measurement Matrix code 1
Extended Kalman Filter Update code 1
Unscented Kalman Filter Update code 1
Main Code 2
Task 7 7

Measurement Matrix code

```
function [measMatrix]= MeasurementMatrix(state,radarState)
% Task 4 - Complete this function
    measMatrix = [ (state(1)-radarState(1)) / (((state(1)-radarState(1))^2
 + (state(2)-radarState(2))^2)^0.5), (state(2)-radarState(2)) / (((state(1)-
radarState(1))^2 + (state(2)-radarState(2))^2)^0.5), 0, 0;
                   ((-state(2)+radarState(2))/((state(2)-radarState(2))^2
 + (state(1)-radarState(1))^2)), ((state(1)-radarState(1))/((state(2)-
radarState(2))^2 + (state(1)-radarState(1))^2)),0 ,0 ];

end
```

Extended Kalman Filter Update code

```
function [xPosterior,PPosterior]= EkfUpdate(xPrior,PPrior,z,R,radarState)
% Task 5 - Complete this function
    zk = [((xPrior(1)-radarState(1))^2 + (xPrior(2)-radarState(2))^2)^0.5 ;
 atan2( (xPrior(2)-radarState(2)) , (xPrior(1)-radarState(1)) ) ];
    HH = MeasurementMatrix(xPrior,radarState);
    S = HH*PPrior*HH' + R;
    W = (PPrior*HH')/S;
    xPosterior = xPrior + W*(z-zk);
    PPosterior = PPrior - W*S*W';
end
```

Unscented Kalman Filter Update code

```
function [xPosterior,PPosterior]= UkfUpdate(xPrior,PPrior,z,R,radarState)
% Task 6 - Complete this function#
    kappa = 0;
```

```
    samplSize = 2*length(xPrior) + 1;
    w = ones(1,samplSize)*(1/(samplSize+kappa));
    w(1) = 0;
    w = w./sum(w);
    sample = zeros(length(xPrior),samplSize);
    hk = zeros(2,samplSize);
    for i = 1:samplSize
        extraElement = real(sqrtm((length(xPrior)+kappa)*PPrior))';
        if i==1
            sample(:,i) = xPrior;
        end
        if i <= (samplSize+1)/2 && i > 1
            sample(:,i) = xPrior + extraElement(:,i-1);
        end
        if i> (samplSize+1)/2
            sample(:,i) = xPrior - extraElement(:,i-length(xPrior)-1);
        end
        hk(:,i) = [((sample(1,i)-radarState(1))^2 + (sample(2,i)-
radarState(2))^2)^0.5 ; atan2( sample(2,i)-radarState(2) , sample(1,i)-
radarState(1) ) ];
    end
    X = sample;
    xk = xPrior;
    zkk = sum(hk.*w,2);
    Pzz = zeros(2,2);
    for i = 1:samplSize
        ele = (hk(:,i)-zkk)*(hk(:,i)-zkk)';
        Pzz = Pzz + ele.*w(i);
    end
    Pxz = zeros(4,2);
    for i = 1:samplSize
        elem = (X(:,i)-xk)*(hk(:,i)-zkk)';
        Pxz = Pxz + elem.*w(i);
    end
    Sk = R + Pzz;
    Kk = Pxz/Sk;
    xPosterior = xPrior + Kk*(z-zkk);
    PPosterior = PPrior - Kk*Sk*Kk';
end
```

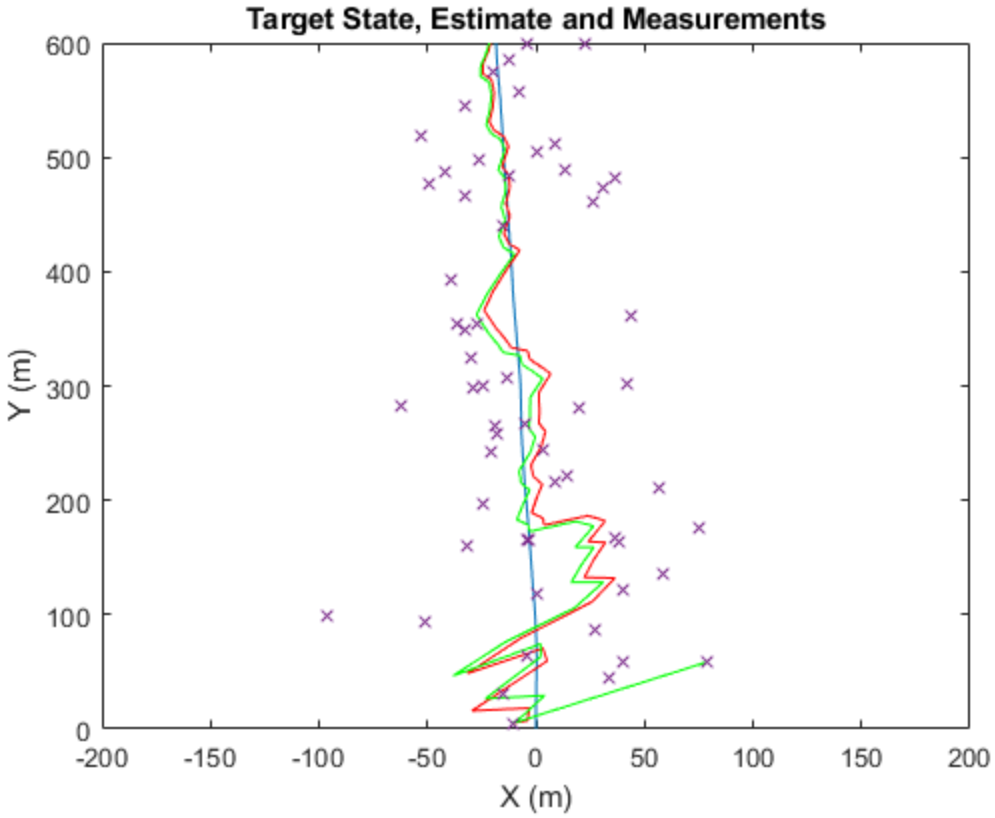
Main Code

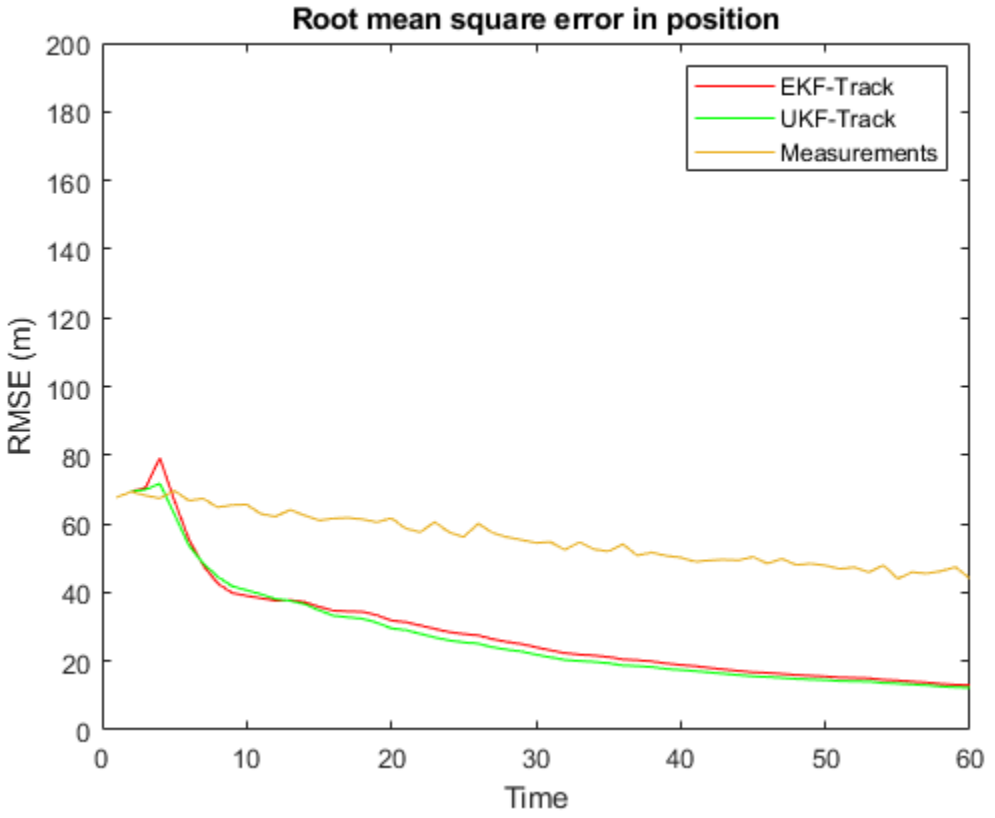
```
close all
clearvars

% Set the number of Monte Carlo runs
numMonteCarlo = 1000;

delT = 1; % Time step
% Task 1 - Complete the state transition matrix
F = [ 1, 0, delT, 0;0, 1, 0, delT;0, 0, 1, 0; 0, 0, 0, 1]; % State transition
 matrix
```

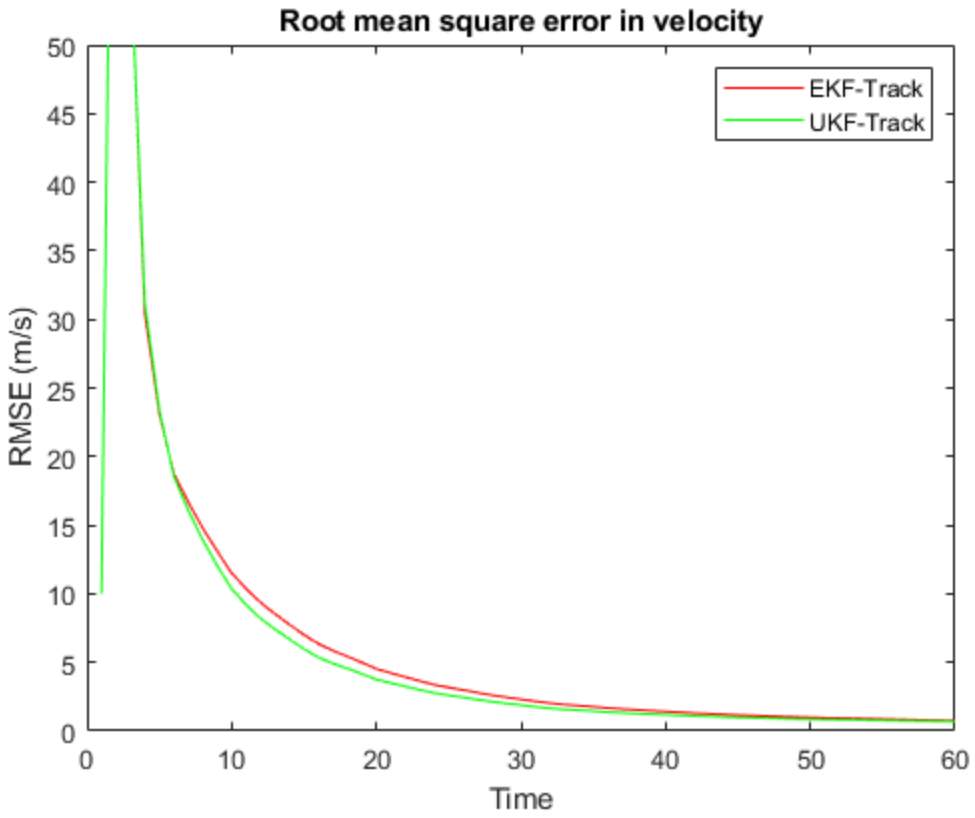
```
proNoise = 0.01; % Process noise
% Task 1 - Complete the state transition noise matrix
Q = proNoise*[delT^3/3, 0, delT^2/2, 0; 0, delT^3/3, 0, delT^2/2; delT^2/2, 0,
 delT, 0; 0, delT^2/2, 0, delT]; % State transition noise matrix

sigmaRange = 5; % Measurement error standard deviation in range
sigmaTheta = 4*pi/180; % Measurement error standard deviation in angle
% Task 2 - Complete the measurement error covariance
R = [sigmaRange^2, 0;0, sigmaTheta^2]; % Measurement error covariance

radarState = [-600 800];

% Preallocate arrays for holding RMSE data
ekfRmseTrackPos = zeros(1,60);
ekfRmseTrackVel = zeros(1,60);
ukfRmseTrackPos = zeros(1,60);
ukfRmseTrackVel = zeros(1,60);
rmseMeasPos = zeros(1,60);

% Loop through the Monte Carlo runs
for j = 1:numMonteCarlo

    % Generate a random target trajectory according to the model
    targetState = zeros(4,60);
    targetState(:,1) = [0 0 0 10]; % Starting state
    for i = 2:60
        % Perform a random walk according to motion model
        targetState(:,i) = F*targetState(:,i-1)+mvnrnd([0 0 0 0]',Q)';
    end

    % Arrays for logging data from a single run
    measurements = zeros(2,60); % Log of measurements in polar space
    measurementsCart = zeros(2,60); % Log of the measurements in Cartesian
 space
    ekfEstimate = zeros(4,60); % Log of EKF estimate
    ukfEstimate = zeros(4,60); % Log of UKF estimate

    % Loop for each time step
    for i = 1:60
        % Generate a random measurement
        z = GenerateMeasurement(targetState(:,i),R,radarState);
        % Store the measurement
        measurements(:,i) = z;
        [measX,measY] = pol2cart(z(2),z(1)); % Convert measurement into
 Cartesian space
        measurementsCart(:,i) = [radarState(1)+measX;radarState(2)+measY]; %
 Store measurement
        if i == 1
            % Store the first time step to be used later in two point
            % initialisation
            ekfEstimate(:,i) = [radarState(1)+measX; radarState(2)+measY;0;0];
            ukfEstimate(:,i) = [radarState(1)+measX; radarState(2)+measY;0;0];
        elseif i == 2
            % Perform two point initialisation
```

```
            ekfMean =
 [radarState(1)+measX;radarState(2)+measY;radarState(1)+measX-
ekfEstimate(1,i-1);radarState(2)+measY-ekfEstimate(2,i-1)];
            ukfMean =
 [radarState(1)+measX;radarState(2)+measY;radarState(1)+measX-
ukfEstimate(1,i-1);radarState(2)+measY-ukfEstimate(2,i-1)];
            ekfEstimate(:,i) = ekfMean;
            ukfEstimate(:,i) = ukfMean;
            % Covariance initialisation
            T = [
                cos(measurements(2,i)) -
measurements(1,i)*sin(measurements(2,i));
                sin(measurements(2,i))
 measurements(1,i)*cos(measurements(2,i));
                ];
            Rcart = T*R*T'; % Measurement error in Cartesian frame
            ekfCovar = [Rcart(1,1) Rcart(1,2) 0 0; Rcart(2,1) Rcart(2,2) 0 0;
 0 0 2*Rcart(1,1) 0; 0 0 0 2*Rcart(2,2)];
            ukfCovar = [Rcart(1,1) Rcart(1,2) 0 0; Rcart(2,1) Rcart(2,2) 0 0;
 0 0 2*Rcart(1,1) 0; 0 0 0 2*Rcart(2,2)];
        elseif i > 2
            % EKF Prediction
            [ekfPriorMean,ekfPriorCovar] =
 kalmanPrediction(ekfMean,ekfCovar,F,Q);
            % UKF Prediction
            [ukfPriorMean,ukfPriorCovar] =
 kalmanPrediction(ukfMean,ukfCovar,F,Q);
            % Extended Kalman update step
            [ekfMean, ekfCovar] =
 EkfUpdate(ekfPriorMean,ekfPriorCovar,z,R,radarState);
            ekfEstimate(:,i) = ekfMean;
            % Extended Kalman update step
            [ukfMean, ukfCovar] =
 UkfUpdate(ukfPriorMean,ukfPriorCovar,z,R,radarState);
            ukfEstimate(:,i) = ukfMean;
        end
    end
    % Log EKF RMSE
    ekfRmseTrackPos = ekfRmseTrackPos + (ekfEstimate(1,:)-targetState(1,:)).^2
 + (ekfEstimate(2,:)-targetState(2,:)).^2;
    ekfRmseTrackVel = ekfRmseTrackVel + (ekfEstimate(3,:)-targetState(3,:)).^2
 + (ekfEstimate(4,:)-targetState(4,:)).^2;
    % Log UKF RMSE
    ukfRmseTrackPos = ukfRmseTrackPos + (ukfEstimate(1,:)-targetState(1,:)).^2
 + (ukfEstimate(2,:)-targetState(2,:)).^2;
    ukfRmseTrackVel = ukfRmseTrackVel + (ukfEstimate(3,:)-targetState(3,:)).^2
 + (ukfEstimate(4,:)-targetState(4,:)).^2;
    % Log measurement RMSE
    rmseMeasPos = rmseMeasPos + (measurementsCart(1,:)-targetState(1,:)).^2 +
 (measurementsCart(2,:)-targetState(2,:)).^2;
    % Make a plot of a single run for only the single run
    if j==1
        figure
        plot(targetState(1,:),targetState(2,:))
```

```
        hold on
        plot(ekfEstimate(1,:),ekfEstimate(2,:),'r')
        plot(ukfEstimate(1,:),ukfEstimate(2,:),'g')
        plot(measurementsCart(1,:),measurementsCart(2,:),'x')
        ylim([0 600])
        xlim([-200 200])
        xlabel('X (m)')
        ylabel('Y (m)')
        title('Target State, Estimate and Measurements')
    end
end

% Calculate the RMSE
ekfRmseTrackPos = sqrt(ekfRmseTrackPos/numMonteCarlo);
ukfRmseTrackPos = sqrt(ukfRmseTrackPos/numMonteCarlo);
rmseMeasPos = sqrt(rmseMeasPos/numMonteCarlo);
ekfRmseTrackVel = sqrt(ekfRmseTrackVel/numMonteCarlo);
ukfRmseTrackVel = sqrt(ukfRmseTrackVel/numMonteCarlo);

% Plot the RMSE in position
figure
plot(1:60,ekfRmseTrackPos(1:60),'r')
hold on
plot(1:60,ukfRmseTrackPos(1:60),'g')
plot(1:60,rmseMeasPos(1:60))
ylim([0 200])
xlabel('Time')
ylabel('RMSE (m)')
title('Root mean square error in position')
legend('EKF-Track','UKF-Track','Measurements')

% Plot the RMSE in velocity
figure
plot(1:60,ekfRmseTrackVel(1:60),'r')
hold on
plot(1:60,ukfRmseTrackVel(1:60),'g')
ylim([0 50])
xlabel('Time')
ylabel('RMSE (m/s)')
title('Root mean square error in velocity')
legend('EKF-Track','UKF-Track')
```


Target State, Estimate and Measurements
Y (m)
X (m)
600
500
400
300
200
100
0
-200
-150
-100
-50
0
50
100
150
200


Root mean square error in position
RMSE (m)
Time
EKF-Track
UKF-Track
Measurements
200
180
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60



Task 7

By varying the sigma in angle measurement, we can notice a change in the performance of Ekf and Ukf Update. As the angle increases from 2->4->6 degrees, the Ekf update logic becomes unstable because of the angle state which is highly non linear causes the EKF linear approximation to become erratic. Meanwhile the UkF update logic does not get impacted that strongly since it depends on a set of sigma points(even this gets impacted due to sigma points being chosen from higher unstable gaussian curve). This shows, UKF update logic has better use in highly unstable models.

Published with MATLAB® R2021b