ISTANBUL TECHNICAL UNIVERSITY

PROBABILISTIC METHODS IN ROBOTICS

HOMEWORK II

HAKAN AŞIK

518182008

13.10.2019

Main m file

```
clear all
clc
% Dimensions
% n=3;
% m=2;
%
% x_t= n x 1
% covPresent_post, covPresent_prior , R_t = n x n
% z_t= m x 1
% C= m x n
% Q_t= m x m
% u_t= c x 1
% B= n x c
% K_t= n x m

xPresent_post  =[0 0 0];
covPresent_post=[0.01   0     0 ;...
                 0    0.01   0; ...
                 0     0     10000];

xPast_post = xPresent_post;
covPast_post= covPresent_post;

u_Present=[];
for t = 1:10

  [xm, ym, theta] = RobotPose(t);          % real value
  [xPresent_post, covPresent_post] =
ExKalFilt(xPast_post, covPast_post, u_Present , [xm ,
ym, theta]')
  xh = double(xPresent_post(1))
  yh = double(xPresent_post(2))
  thetah = double(xPresent_post(3))

  xPast_post = double(xPresent_post)';
  covPast_post= covPresent_post;

  Xmsaved(t,:) = [xm, ym, theta];
  Xhsaved(t,:) = [xh, yh, thetah];

end
```

```
figure (1)
plot(t,Xmsaved(:,1),'r','linewidth',4)    % x value
(real)
hold on
plot(t,Xhsaved(:,1),'b','linewidth',4)  % x value
(estimation)
xlabel('time', 'FontSize', 24);
ylabel('x values', 'FontSize', 24);
legend('"x" position','"x" estimation')
set(gca,'FontSize',24,'fontWeight','bold')
grid
%
figure (2)
plot(t,Xmsaved(:,2),'r','linewidth',4)    % y value
(real)
hold on
plot(t,Xhsaved(:,2),'b','linewidth',4)  % y value
(estimation)
xlabel('time', 'FontSize', 24);
ylabel('y values', 'FontSize', 24);
legend('"y" position','"y" estimation')
set(gca,'FontSize',24,'fontWeight','bold')
grid
%
figure (3)
plot(t,Xmsaved(:,3),'r.','linewidth',4)   % theta value
(real)
hold on
plot(t,Xhsaved(:,3),'b.','linewidth',4)   % theta value
(estimation)
xlabel('time', 'FontSize', 24);
ylabel('theta values', 'FontSize', 24);
legend('"theta" position','"theta" estimation')
set(gca,'FontSize',24,'fontWeight','bold')
grid
%
figure (4)
plot(Xmsaved(:,1),Xmsaved(:,2),'r.','linewidth',4)    %
real values in x,y direction
hold on
plot(Xhsaved(:,1),Xhsaved(:,2),'b.','linewidth',4)%
estimation values in x,y direction
xlabel('x values', 'FontSize', 24);
ylabel('y values', 'FontSize', 24);
```

```
legend('real orbit','estimation orbit')
set(gca,'FontSize',24,'fontWeight','bold')
grid
```

Robot Pose m file

```
function [xm, ym, theta] =RobotPose(t)
%
persistent Posxm Posym

if isempty(Posxm)
    Posxm=0;
    Posym=0;
end

d=1;
%% Generate multivariate values from a normal
distribution with specified mean vector and covariance
matrix.
mu = [0 0 0];
sigma = [0.01 0 0; 0 0.01 0; 0 0 10000];
R = chol(sigma);
z=mvnrnd(mu,sigma);
theta=z(1,3);
%%
xm=Posxm+d*cosd(theta);
ym=Posym+d*sind(theta);

Posxm=xm;          % true X position
Posym=ym;          % true Y position
```

Extended Kalman Filter function m file

```
function [xPresent_post, covPresent_post] =
ExKalFilt(xPast_post, covPast_post, u_Present,
z_Present)
% gFunc: Dynamic model-state transition matrix
% hFunc: Observation model- measurement function
% R_t   : Dynamical model Gaussian noise covariance
% Q_t   : Observation model Gaussian noise covariance


% Dimensions
% n=3;
% m=3;
%
% x_t= n x 1
% covPresent_post, covPresent_prior , R_t = n x n
% z_t= m x 1
% C= m x n
% Q_t= m x m
% u_t= c x 1
% B= n x c
% K_t= n x m

muR_t=zeros(3,3);
sigmaR_t=[0.01    0      0 ;...
           0     0.01    0; ...
           0      0     10000];
R_t = normrnd(muR_t,sigmaR_t)

muQ_t=zeros(3,3);
sigmaQ_t=[0.01 0 0 ;0 0 0; 0 0 0];
Q_t = normrnd(muQ_t,sigmaQ_t)

% Dynamical Model
d=1;

 syms xPast_post1 xPast_post2 xPast_post3
gFunc= [xPast_post1+d* cosd(xPast_post3);...% g1= X'
        xPast_post2+d* sind(xPast_post3);... % g2= Y'
            xPast_post3]  % - 3x1
```

```
%Calculate G_t; Jacobian matrix of gFunc
Gg=jacobian(gFunc,[xPast_post1,xPast_post2,xPast_post3]
)
G_t=
double(subs(Gg,[xPast_post1,xPast_post2,xPast_post3],{x
Past_post(1), xPast_post(2), xPast_post(3)}))

%% Prediction Step
%   Step I: Compute State prediction
xPresent_prior=transpose(double(subs(gFunc,[xPast_post1
,xPast_post2,xPast_post3],{xPast_post(1),
xPast_post(2), xPast_post(3)}))) % 3x1
%   Step II: Compute Covariance prediction
covPresent_prior= G_t * covPast_post * (G_t)' + R_t ; %
- 3x3 * 3x3  * 3x3 + 3x3 = 3x3

% Observation Model
  H = [ 1 0 0;
        0 1 0;
        0 0 1];

x_t=xPresent_prior(1) ; y_t=xPresent_prior(2) ;
theta=xPresent_prior(3)  ; x_s=xPast_post(1) ;
y_s=xPast_post(2) ;
hFuncVal= [sqrt((x_t-x_s)^2+(y_t-
y_s)^2)*cosd(theta),sqrt((x_t-x_s)^2+(y_t-
y_s)^2)*sind(theta) ,atand((y_s-y_t)/(x_s-x_t)-theta)]
% Observation model - 3x1

H_tVal= [x_t/sqrt(x_t^2+y_t^2)
y_t/sqrt(x_t^2+y_t^2)   -sind(theta); ...
         x_t/sqrt(x_t^2+y_t^2)
y_t/sqrt(x_t^2+y_t^2)   cosd(theta);...
        -y_t/(x_t^2+y_t^2)        x_t/(x_t^2+y_t^2)
0] % mxn - 3x3

%% Correction Step
%   Step III: Compute Kalman gain (K_t)
K_t= covPresent_prior * (H_tVal)' * inv(H_tVal *
covPresent_prior * (H_tVal)' + Q_t) % 3x3 * 3x3 * inv(
2x3 * 3x3 * 3x3 + 3x3) = 3x3
%   Step IV:  Compute State Estimate
xPresent_post= double(xPresent_prior') + K_t *
(z_Present - H * xPresent_prior')
```

```
%   Step V:   Compute Covariance Estimate
covPresent_post= (eye(3) - K_t * H_tVal) *
covPresent_prior % ( 3x3 - 3x3 * 3x3) * 3x3 = 3x3

end
```