clear all

% duration and how often we sample

duration = 10; %car ride duration

dt = .1; % sampling distance

% Define update equations

Fk = [1 dt; 0 1] ; %State Transition Matrix

Bk = [dt^2/2; dt]; %Input Control Matrix

Hk = [1 0]; % Measurement matrix

%we are only measuring position, so velocity variable is set to zero.

% main variables

u = 1.5; % acceleration mag

x= [0; 0]; %initial state vector, car has two components: [position; velocity]

xhat = x; %initial state estimation of where the car is (what we are updating)

car\_accel\_noise\_mag = 0.05; %process noise -standard deviation of acceleration

robot\_noise\_mag = .10; %measurement noise -standard deviation of location

sigmaw = car\_accel\_noise\_mag^2 \* [dt^4/4 dt^3/2; dt^3/2 dt^2]; % Process noise covariance matrix

Rk = robot\_noise\_mag^2;% measurement noise covariance matrix

Pk = sigmaw; % initial estimation of car position covariance

% result variables

pos = []; % Actual car ride trajectory

vel = []; % Actual car velocity

Zk = []; % car trajectory that the robot sees (measured) robots perception

% simulate what robot sees over time

for t = 0 : dt: duration

% Generate the car ride

processNoise = car\_accel\_noise\_mag \* [(dt^2/2)\*randn; dt\*randn];

x= Fk \* x+ Bk \* u + processNoise;

% Generate what the robot sees

measurementNoise = robot\_noise\_mag \* randn\*100;

y = Hk \* x+ measurementNoise;

pos = [pos; x(1)];

Zk = [Zk; y];

vel = [vel; x(2)];

end

% Plot the results

figure(1);

tt1=0:dt:t;

% Actual ride of car % what robot sees contineously %theoretical trajectory of robot that doesn't use kalman,but using moving average summing in window

plot(tt1, pos, '-r.',tt1, Zk, '-k.',tt1, smooth(Zk), '-g.'),title ('without kalman filter'),

axis([0 10 -20 80]),legend('Actual trajectory of car','what robot sees','estimate' );

% using kalman filtering

% estimation variables

pos\_estimate = []; % car position estimate

vel\_estimate = []; % car velocity estimate

x= [0; 0]; % reinitialize the state

P\_mag\_estimate = [];

predict\_state = [];

predict\_var = [];

for t = 1:length(pos)

% Predict next state of the car with the last state and predicted motion.

xhat = Fk \* xhat + Bk \* u;

predict\_state = [predict\_state; xhat(1)] ;

%predict next covariance

Pk = Fk \* Pk \* Fk' + sigmaw;

predict\_var = [predict\_var; Pk] ;

% predicted robot measurement covariance

% Kalman Gain

K = Pk\*Hk'\*inv(Hk\*Pk\*Hk'+Rk);

% Update the state estimate.

xhat = xhat + K \* (Zk(t) - Hk \* xhat);

% update covariance estimation.

Pk = (eye(2)-K\*Hk)\*Pk;

%Store result for plotting

pos\_estimate = [pos\_estimate; xhat(1)];

vel\_estimate = [vel\_estimate; xhat(2)];

P\_mag\_estimate = [P\_mag\_estimate; Pk(1)];

end

% Plot the results

figure(2);

tt2 = 0 : dt : duration;

plot(tt2,pos,'-r.',tt2,Zk,'-k.', tt2,pos\_estimate,'-g.'),title ('with kalman filter'),

axis([0 10 -20 80]),legend('Actual trajectory of car','what robot sees','kalman filter estimate' );

%plot the evolution of the distributions

figure(3);

for T = 1:length(pos\_estimate)

clf

x = pos\_estimate(T)-5:.01:pos\_estimate(T)+5; % x axis range

%predicted next position of the car

hold on

mu = predict\_state(T); % mean

sigma = predict\_var(T); % standard deviation

y = normpdf(x,mu,sigma); % pdf

y = y/(max(y));

hl = line(x,y,'Color','m');

%data measured by the robot

mu = Zk(T); % mean

sigma = robot\_noise\_mag; % standard deviation

y = normpdf(x,mu,sigma); % pdf

y = y/(max(y));

hl = line(x,y,'Color','k'); % or use hold on and normal plot

%combined position estimate

mu = pos\_estimate(T); % mean

sigma = P\_mag\_estimate(T); % standard deviation

y = normpdf(x,mu,sigma); % pdf

y = y/(max(y));

hl = line(x,y, 'Color','g');

axis([pos\_estimate(T)-5 pos\_estimate(T)+5 0 1]);

%actual position of the car

plot(pos(T));

ylim=get(gca,'ylim');

line([pos(T);pos(T)],ylim.','linewidth',2,'color','b');

legend('state predicted','measurement','state estimate','actual car position')

pause

end