```
%=========================================================================
% ME 652, Spring 2020
% Course instructor: Jinwhan Kim
%=========================================================================

function [] = SLAM_skeleton(noiseLevel)
randn('seed',1);
DTR = pi/180;   % degree to radian

% true initial condition
x = [1; 1; 45*DTR; 1;   % x, y, psi, V
     4.6454;  8.0242;  % landmark 1 (x1,y1)
     7.5198;  4.7523;  % landmark 2 (x2,y2)
     1.6836;  5.4618;  % landmark 3 (x3,y3)
     5.5984;  3.1042;  % landmark 4 (x4,y4)
     8.4626;  1.6814]; % landmark 5 (x5,y5)

% initial error covariance
Phat = eye(14);
Phat(1,1) = 0;
Phat(2,2) = 0;

% process noise
Qe = zeros(14,14);

% measurement noise
Re = eye(12);
if nargin == 0
    Re(1,1) = (3*DTR)^2;
    Re(2,2) = 0.2^2;
    for i=1:5
        Re(i*2+1,i*2+1) = (3*DTR)^2;
        Re(i*2+2,i*2+2) = 0.2^2;
    end
else
    Re(1,1) = (3*noiseLevel*DTR)^2;
    Re(2,2) = (0.2*noiseLevel)^2;
    for i=1:5
        Re(i*2+1,i*2+1) = (3*noiseLevel*DTR)^2;
        Re(i*2+2,i*2+2) = (0.2*noiseLevel)^2;
    end
end

% initial estimate
xhat = x;
xhat(5:14) = x(5:14) + sqrt(Phat(5:14,5:14))*randn(10,1);

figure;
set(gcf,'position',[100 100 500 500])

dt = 0.1;   % simulation time step
for t=0:dt:10
    x = propagate_x(x,dt);  % true state propagation
% -----------------------------------------------------------------------
%    Implement the following functions
% -----------------------------------------------------------------------
   z = generate_z(x);      % generate bearing and range measurements
   [xhat,Phat] = EKF_propagate(xhat,Phat,Qe,dt);  % state propagation
   [xhat,Phat] = EKF_update(xhat,z,Phat,Re,dt);   % measurement update
```

```
% ------------------------------------------------------------------------
    clf
    show_data(t,x,xhat,Phat);
    drawnow;
end

function z = generate_z(x)
%     disp('starting measurements ...')
    z(1) = x(3); % vehicle bearing
    z(2) = x(4); % vehicle speed
    for i = 1:5
        % bearing
        z(2*i+1) = atan2( x(2*i+4)-x(2), x(2*i+3) -x(1))-x(3);
        % range
        z(2*i+2) = norm( x(2*i+3:2*i+4) - x(1:2) );
    end

function  [xhat,Phat] = EKF_propagate(xhat,Phat, Qe, dt)
%     disp('starting propagation ...')
    Q = Qe*dt;
    % A Jacobian
    A = jacob_A(xhat);

    Ak = eye(14)+A*dt;
    Phat = Ak*Phat*Ak' + Q ;
    xhat = propagate_x(xhat,dt);

function A = jacob_A(xhat)
    A = zeros(14);
    A(1,3) = - xhat(4)*sin(xhat(3));
    A(2,3) = xhat(4)*cos(xhat(3));
    A(1,4) = cos(xhat(3));
    A(2,4) = sin(xhat(3));

function [xhat,Phat] = EKF_update(xhat,z,Phat,Re,dt)
%     disp('starting update ...')
    R = Re/dt;
    % H Jacobian
    H = jacob_H(xhat);
    % Kalman Gain
    K = Phat*H'/(H*Phat*H'+R);
    % update steps
    zhat = generate_z(xhat);
    Phat = (eye(14) - K*H)*Phat;
    xhat = xhat + K*(z - zhat)';

function H = jacob_H(xhat)
    H = zeros(12,14);
    H(1,3) = 1;
    H(2,4) = 1;
    for i = 1:5
        % xx = x_i - x_0
        xx = xhat(2*i+3) - xhat(1);
        % yy = y_i - y_0
        yy = xhat(2*i+4) - xhat(2);
        % rr = sqrt(xx^2 + yy^2)
        rr = sqrt(xx^2 + yy^2);

        H(2*i+1,1) = yy/rr^2;
        H(2*i+1,2*i+3) = -yy/rr^2;
        H(2*i+1,2) = -xx/rr^2;
        H(2*i+1,2*i+4) = xx/rr^2;
        H(2*i+2,1) = -xx/rr;
        H(2*i+2,2*i+3) = xx/rr;
        H(2*i+2,2) = -yy/rr;
        H(2*i+2,2*i+4) = yy/rr;
    end

% state propagation
%--------------------------------------------------------------
function x = propagate_x(x,dt)
xdot = zeros(14,1);
xdot(1) = x(4)*cos(x(3));   % xdot
xdot(2) = x(4)*sin(x(3));   % ydot
xdot(3) = 0;    % yaw rate
xdot(4) = 0;    % speed
x = x + xdot*dt;    % time integration (Euler scheme)


% show data
%--------------------------------------------------------------
function show_data(t,x,xhat,Phat)
hold on

L = 0.6; % vehicle length
B = 0.3;    % vehicle breadth
objx = L*[1 -1 -1 1]';
objy = B*[0,1,-1,0]';

posx = x(1) + objx*cos(x(3))-objy*sin(x(3));
posy = x(2) + objx*sin(x(3))+objy*cos(x(3));
fill(posx,posy,'y');
plot(posx,posy,'b');

posx = xhat(1) + objx*cos(xhat(3))-objy*sin(xhat(3));
posy = xhat(2) + objx*sin(xhat(3))+objy*cos(xhat(3));
plot(posx,posy,'r');

plot(0,0,'bo',x(1),x(2),'bo',xhat(1),xhat(2),'r+');
meanval = [xhat(1),xhat(2)];
sigma_u = Phat(1:2,1:2);

confidence = 0.95;  % confidence level
alpha = chi2inv(confidence,2);
plot2Dellipse(alpha,meanval,sigma_u,'r');

for i=1:5
    plot(xhat(i*2+3),xhat(i*2+4),'r+',x(i*2+3),x(i*2+4),'k+');
    meanval = [xhat(i*2+3),xhat(i*2+4)];
    sigma_u = Phat(i*2+3:i*2+4,i*2+3:i*2+4);
    plot2Dellipse(alpha,meanval,sigma_u,'r');
end

s = sprintf('Time = %8.2f',t);
text(1,9,s,'fontsize',15);
xlabel('x','fontsize',15), ylabel('y','fontsize',15);
set(gca,'fontsize',15,'PlotBoxAspectRatio',[500 500 100]);
set(gca, 'Ydir','reverse');
```

```
axis([0 10 0 10]);
grid on; box on;

% draw a confidence ellipse
%-----------------------------------------------------------
function [] = plot2Dellipse(alpha,meanval,sigma,color)
[V,D] = eig(sigma);

a = sqrt(alpha*D(1,1));
b = sqrt(alpha*D(2,2));

ang1 = atan2(V(2,1),V(1,1));

theta = (0:360)/180*pi;
x = a*cos(theta);
y = b*sin(theta);
xn = x*cos(ang1)-y*sin(ang1)+meanval(1);
yn = x*sin(ang1)+y*cos(ang1)+meanval(2);

plot(xn,yn,color,'LineWidth',2);
```