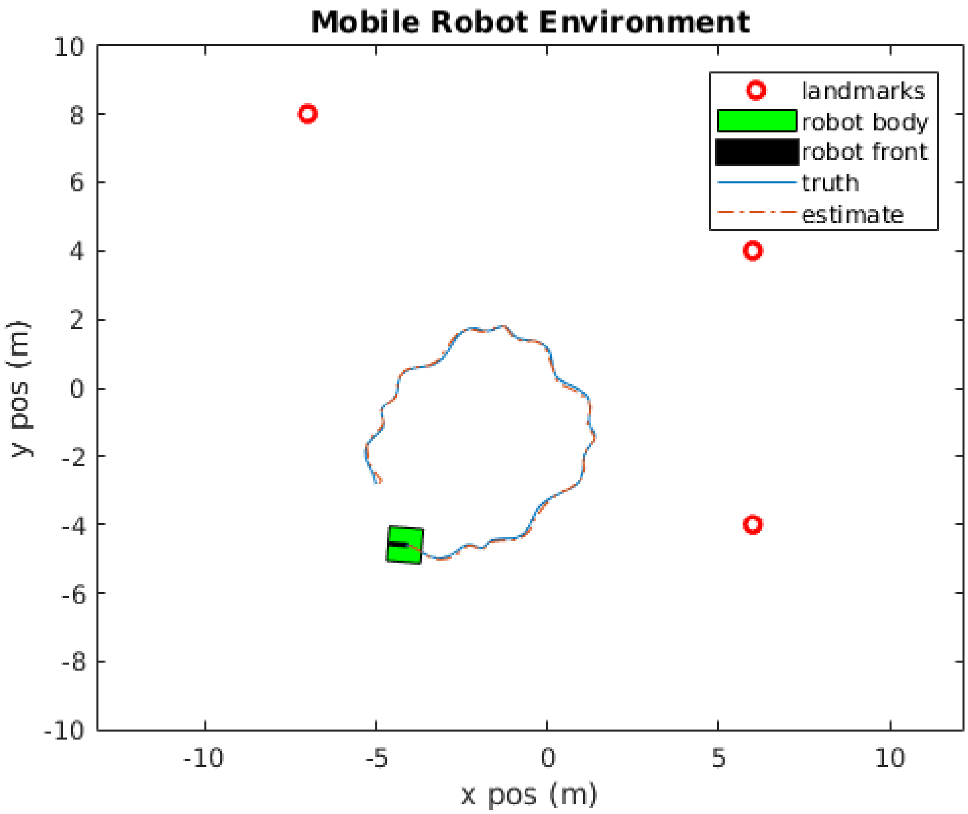
```
% Homework 3 Autonomous Two-wheel robot UKF
% Jesse Wynn
% October 03, 2017

clc
clear
close all

% time params
Ts = 0.1;   % sec
t = 0:Ts:20;

% commanded velocity
v_c = 1 + 0.5 * cos(2 * pi * (0.2) * t);
w_c = -0.2 + 2 * cos(2 * pi * (0.6) * t);

% noise params
alpha_1 = 0.1;
alpha_4 = 0.1;
alpha_2 = 0.01;
alpha_3 = 0.01;
alpha_5 = 0;
alpha_6 = 0;

sigma_r = 0.1;
sigma_phi = 0.05;

% landmark locations
landmarks = [6, -7 6;   % [x_positions; y_positions]
             4, 8, -4];
% landmarks = landmarks(:,1); % just one landmark for now

num_landmarks = size(landmarks);
num_landmarks = num_landmarks(2);

% allocate space for holding data
x_true = zeros(1,length(t));
y_true = zeros(1,length(t));
theta_true = zeros(1,length(t));

range = zeros(length(t),num_landmarks);
bearing = zeros(length(t),num_landmarks);

x_est = zeros(1,length(t));
y_est = zeros(1,length(t));
theta_est = zeros(1,length(t));

Sigma_x = zeros(1,length(t));
Sigma_y = zeros(1,length(t));
Sigma_theta = zeros(1,length(t));
```

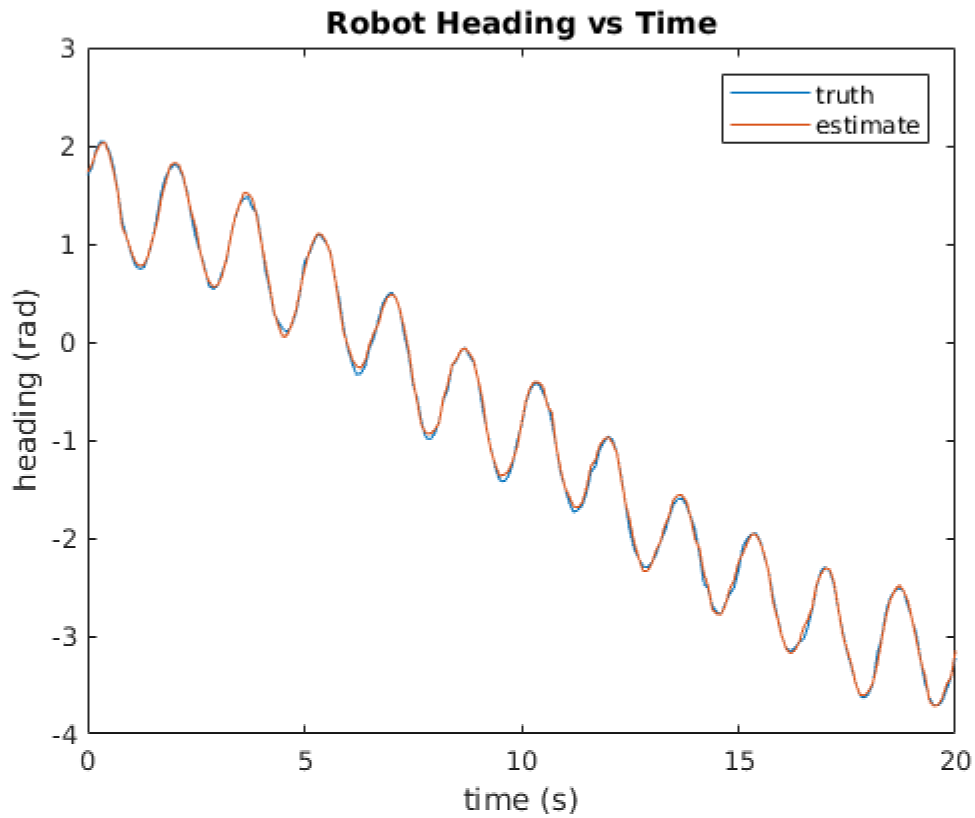
```
% initital conditions
x = -5;
y = -3;
theta = pi/2;
first = 0;
a = 0;

% initial state
mu_t = [x; y; theta];
Sigma_t = [0.1, 0, 0;
           0, 0.1, 0;
           0, 0, 0.1];

% tuning params for sigma points
k = 1;
alpha = 1;
beta = 2;

% size of the augmented state
n = 7;

% size of the state
num_states = 3;

lambda = alpha^2 * (n + k) - n;
gamma = sqrt(n + lambda);

% find your weights
weight_m = zeros(1, 2*n + 1);   % weight of the mean
weight_c = zeros(1, 2*n + 1);   % weight of the covariance

weight_m(1) = lambda/(n + lambda);
weight_m(2:length(weight_m)) = 1/(2*(n + lambda));

weight_c(1) = lambda/(n + lambda) + (1 - alpha^2 + beta);
weight_c(2:length(weight_c)) = 1/(2*(n + lambda));

landmark_number = 1;    % start with the first landmark

% plot the first time
drawRobot(x,y,theta,landmarks,first);
first = 1;
for i = 1:length(t)

    pause(0.01)
    % implement velocity motion model from Table 5.3 (this gives us
 truth)
    v_hat = v_c(i) + randn*sqrt(alpha_1*(v_c(i))^2 +
 alpha_2*(w_c(i))^2);
    w_hat = w_c(i) + randn*sqrt(alpha_3*(v_c(i))^2 +
 alpha_4*(w_c(i))^2);
    gamma_hat = randn*sqrt(alpha_5*(v_c(i))^2 + alpha_6*(w_c(i))^2);
    x = x - (v_hat/w_hat)*sin(theta) + (v_hat/w_hat)*sin(theta +
 w_hat*Ts);
```

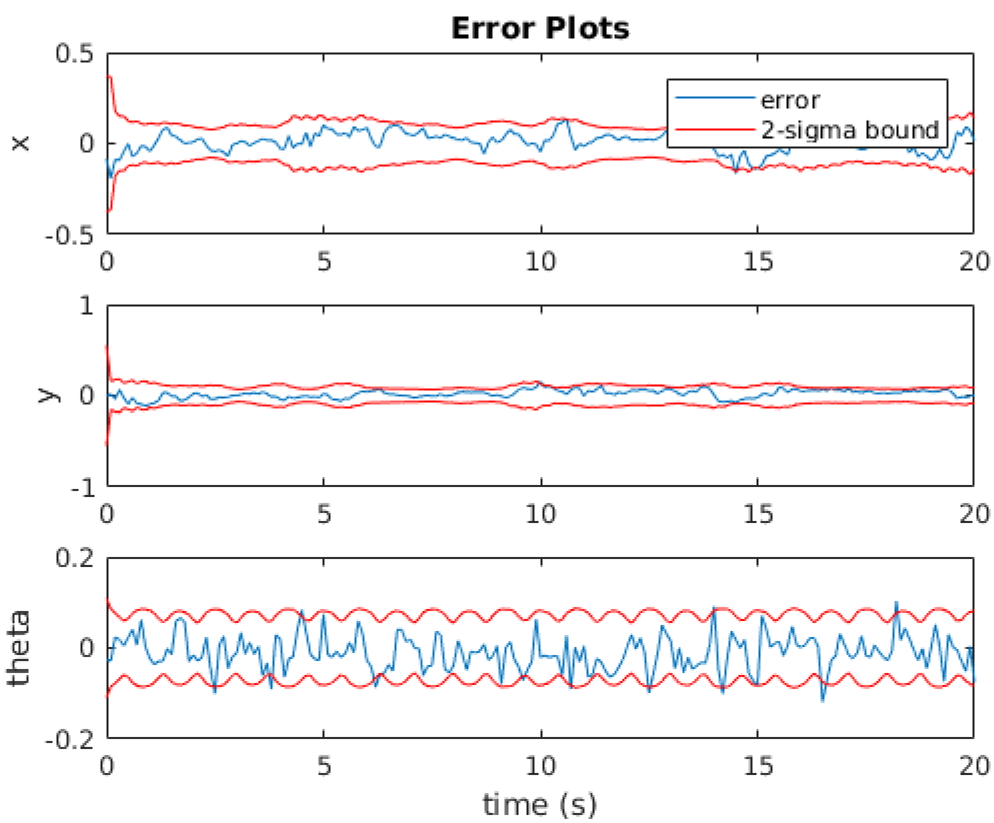
```
   y = y + (v_hat/w_hat)*cos(theta) - (v_hat/w_hat)*cos(theta +
w_hat*Ts);
   theta = theta + w_hat*Ts + gamma_hat*Ts;

   % update the plot
   drawRobot(x,y,theta,landmarks,first)

   % save some data for plotting
   x_true(i) = x;
   y_true(i) = y;
   theta_true(i) = theta;

   % simulate range and bearing measurements to landmarks
   range(i,1) = norm([x; y] - landmarks(:,1)) + randn*sigma_r;   %
measurement + noise
   range(i,2) = norm([x; y] - landmarks(:,2)) + randn*sigma_r;
   range(i,3) = norm([x; y] - landmarks(:,3)) + randn*sigma_r;

   bearing(i,1) = atan2(landmarks(2,1) - y, landmarks(1,1) - x) -
theta + randn*sigma_phi;  % measurement + noise
   bearing(i,2) = atan2(landmarks(2,2) - y, landmarks(1,2) - x) -
theta + randn*sigma_phi;   % maybe sqrt
   bearing(i,3) = atan2(landmarks(2,3) - y, landmarks(1,3) - x) -
theta + randn*sigma_phi;


   % implement the UKF algorithm found in Table 7.4 (p. 221)

   % inuput velocities
   u = [v_c(i); w_c(i)];
   v_t = u(1);
   w_t = u(2);

   % line 2
   M_t = [alpha_1 * v_t^2 + alpha_2 * w_t^2, 0; 0, alpha_3 * v_t^2 +
alpha_4 * w_t^2];

   % line 3
   Q_t = [sigma_r^2, 0;
          0, sigma_phi^2];

   % line 4
   mu_t_aug = [mu_t; 0; 0; 0; 0;]; % augmented state

   % line 5
   Sigma_t_aug = [Sigma_t, zeros(3,2), zeros(3,2); % augmented
covariance
                 zeros(2,3), M_t, zeros(2,2);
                 zeros(2,3), zeros(2,2), Q_t];

   % generate sigma points
   % line 6
```

```
   Chi_t_aug = [mu_t_aug, mu_t_aug + gamma * chol(Sigma_t_aug),
mu_t_aug - gamma * chol(Sigma_t_aug)]; % chol is the cholesky
factorization

   % pass sigma points through motion model and compute gaussian
stats
   % line 7
   Chi_bar_t_x = g(u, Chi_t_aug, Ts);  % NOT 100% sure I'm passing
the right stuff here

   % line 8
   mu_t =  Chi_bar_t_x * weight_m';

   % line 9
   Sigma_t = weight_c .* (Chi_bar_t_x - mu_t) * (Chi_bar_t_x -
mu_t)';

   % predict observations at sigma points and compute Gaussian stats

   % cycle through landmarks but use only one each time through the
loop:
   % line 10
   Z_bar_t = h([landmarks(1,landmark_number),
landmarks(2,landmark_number)]', Chi_bar_t_x, Chi_t_aug(6:7,:));

   % line 11
   z_hat_t = Z_bar_t * weight_m';

   % line 12
   S_t = weight_c .* (Z_bar_t - z_hat_t) * (Z_bar_t - z_hat_t)';

   % line 13
   Sigma_t_x_z = weight_c .* (Chi_bar_t_x - mu_t) * (Z_bar_t -
z_hat_t)';

   % update mean and covariance
   % line 14
   K_t = Sigma_t_x_z / S_t;

   % line 15
   mu_t = mu_t + K_t * ([range(i,landmark_number);
bearing(i,landmark_number)] - z_hat_t);

   % line 16
   Sigma_t = Sigma_t - K_t * S_t * K_t';

   % next time through, cycle to the next landmark
   landmark_number = landmark_number + 1;

   if landmark_number > num_landmarks
       landmark_number = 1;     % reset
   end

   % End UKF implementation
```

```
    % save off some stuff for plotting
    x_est(i) = mu_t(1);
    y_est(i) = mu_t(2);
    theta_est(i) = mu_t(3);

    Sigma_x(i) = Sigma_t(1,1);
    Sigma_y(i) = Sigma_t(2,2);
    Sigma_theta(i) = Sigma_t(3,3);
end

plot(x_true, y_true, x_est, y_est,'-.')
legend('landmarks','robot body','robot front','truth','estimate')

figure(2), clf
plot(t,theta_true,t,theta_est)
title('Robot Heading vs Time')
xlabel('time (s)')
ylabel('heading (rad)')
legend('truth','estimate')

figure(3)
subplot(3,1,1)
plot(t,x_true - x_est,t,2*sqrt(Sigma_x),'r',t,-2*sqrt(Sigma_x),'r')
title('Error Plots')
ylabel('x')
legend('error','2-sigma bound')

subplot(3,1,2)
plot(t,y_true - y_est,t,2*sqrt(Sigma_y),'r',t,-2*sqrt(Sigma_y),'r')
ylabel('y')

subplot(3,1,3)
plot(t,theta_true -
 theta_est,t,2*sqrt(Sigma_theta),'r',t,-2*sqrt(Sigma_theta),'r')
ylabel('theta')
xlabel('time (s)')
```


Mobile Robot Environment
landmarks
robot body
robot front
truth
estimate
y pos (m)
x pos (m)
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10
-5
0
5
10


Robot Heading vs Time
truth
estimate
heading (rad)
time (s)


Error Plots
error
2-sigma bound
x
y
theta
time (s)

Published with MATLAB® R2017a