MAE 6292: Final Exam

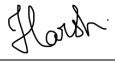
Due: 5pm, Friday, May 8 2020

Uppaluru	, Harshvardhan	. G42420576
Last Name	First Name	Student ID

Prob. 1	Prob. 2	Prob. 3	Prob. 4	Total

Honor Pledge

I pledge that I have neither given nor received unauthorized assistance on this work. I understand that any academic misconduct will be handled according to *GWU Code of Academic Integrity*.



5/8/2020

Signature Date

Note

- Make all of your submission into a single pdf file, organized as
  - Cover page
  - Honor pledge signed by you
  - Written answer for Question 1
  - Matlab code and results for Question 1
  - Written answer for Question 2
  - Matlab code and results for Question 2
  - ⋮
- Use the *publish* feature of the Matlab to share the code and the results:
  `https://www.mathworks.com/help/matlab/matlab_prog/publishing-matlab-code.html`

Problem 1 (EKF for Localization) Consider a planar robot model whose configuration is described by its location (x, y) and the orientation θ. It translates with a fixed velocity V along its heading-direction that can be changed by a control input u, i.e., it may represent a wheeled planar robot. Assume that the velocity and the control input are perturbed by process noise w_{v_k}, w_{θ_k} respectively. The equations of motion are given by

$$\begin{aligned} x_{k+1} &= x_k + h(V + w_{v_k})\cos\theta_k, \\ y_{k+1} &= y_k + h(V + w_{v_k})\sin\theta_k, \\ \theta_{k+1} &= \theta_k + h(u_k + w_{\theta_k}), \end{aligned}$$

where h denotes the time step.

Consider several reference points around the robot. Suppose that the location of the i-th reference point is known, and it is given by (r_x^i, r_y^i). The robot is equipped with a range sensor to measure the distance between the robot and the reference point, i.e., the i-th range measurement is given by

$$z_r^i = \sqrt{\Delta x_i^2 + \Delta y_i^2} + v_r,$$

where the relative position of the i-th reference point from the robot is denoted by $(\Delta x_i, \Delta y_i) = (r_x^i - x, r_y^i - y)$, and v_r corresponds to the range measurement noise.

The robot is also equipped with a bearing sensor to measure the direction toward reference points. Since the bearing sensor is attached to the body of the robot, the sensor reading is given with respect to the body-fixed frame. The relative position is represented with respect to the body-fixed frame as

$$\begin{bmatrix} \Delta x_i^{body} \\ \Delta y_i^{body} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix}.$$

The corresponding i-th bearing measurement is

$$z_\theta^i = \tan^{-1}\frac{\Delta y_i^{body}}{\Delta x_i^{body}} + v_\theta,$$

where v_i denotes the measurement error. The complete measurement is given by $\mathbf{z} = [z_r^1, z_\theta^1, \ldots, z_r^{N_r}, z_\theta^{N_r}] \in \mathbb{R}^{2N_r}$ where N_r denotes the number of reference points.

(a) Let the state vector be $\mathbf{x}_k = [x_k, y_k, \theta_k]^T \in \mathbb{R}^3$. Show that the linearized equations of motion are given by

$$\delta\mathbf{x_{k+1}} = A_k\delta\mathbf{x_k} + B_k u_k + G_k w_k,$$

where

$$A_k = \begin{bmatrix} 1 & 0 & -hV\sin\theta_k \\ 0 & 1 & hV\cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}, \quad B_k = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}, \quad G_k = \begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \end{bmatrix}.$$

(Note: the increase of P_k due to the process noise becomes $G_k Q_k G_k^T$.)

(b) Let the measurements from the i-th reference point be $\mathbf{z}^i = [z_r^i, z_\theta^i]^T \in \mathbb{R}^2$. Show that the linearized measurement equation can be written as

$$\delta\mathbf{z}^i = H^i\delta\mathbf{x},$$

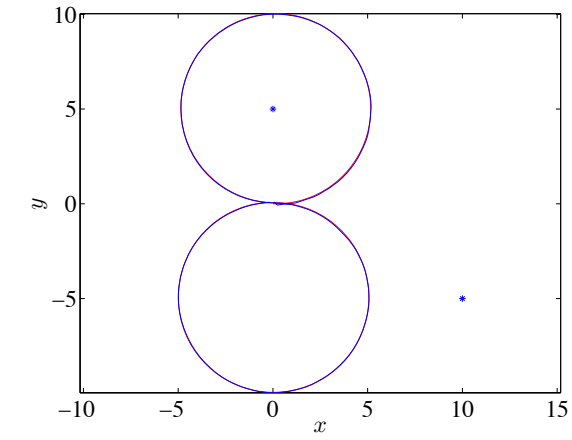
where the matrix $H^i \in \mathbb{R}^{2\times 3}$ are given by

$$H^i = \begin{bmatrix} \frac{\partial z_r^i}{\partial x} & \frac{\partial z_r^i}{\partial y} & \frac{\partial z_r^i}{\partial \theta} \\ \frac{\partial z_\theta^i}{\partial x} & \frac{\partial z_\theta^i}{\partial y} & \frac{\partial z_\theta^i}{\partial \theta} \end{bmatrix},$$

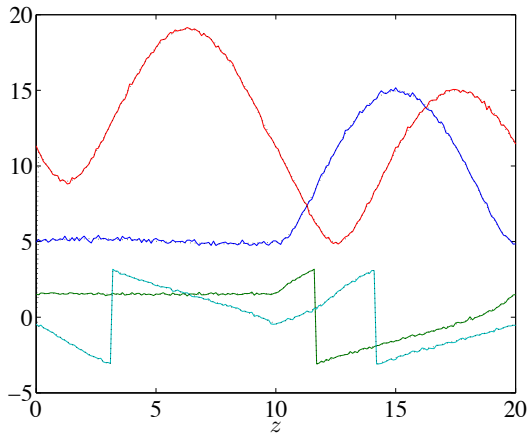
and the derivatives are given by

$$\frac{\partial z_r^i}{\partial x} = -\frac{\Delta x_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}}, \quad \frac{\partial z_r^i}{\partial y} = -\frac{\Delta y_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}}, \quad \frac{\partial z_r^i}{\partial \theta} = 0,$$
$$\frac{\partial z_\theta^i}{\partial x} = \frac{\Delta y_i}{\Delta x_i^2 + \Delta y_i^2}, \quad \frac{\partial z_\theta^i}{\partial y} = -\frac{\Delta x_i}{\Delta x_i^2 + \Delta y_i^2}, \quad \frac{\partial z_\theta^i}{\partial \theta} = -1.$$

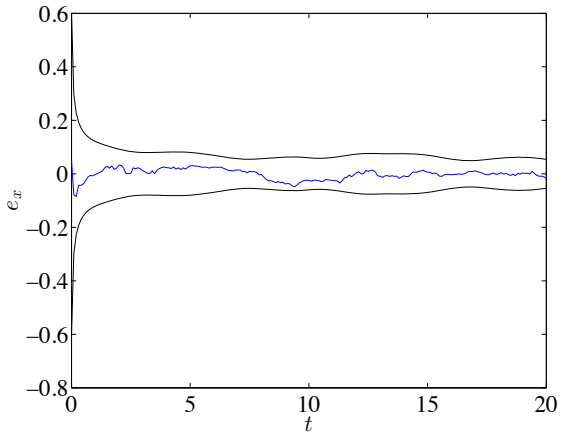
(c) All of the parameters for the robot model, control inputs, measurements, reference points, and initial guess are specified at `prob1.m`. Write a Matlab file for EKF to estimate the state $\mathbf{x}_k$, and generate 6 plots given at the next page.



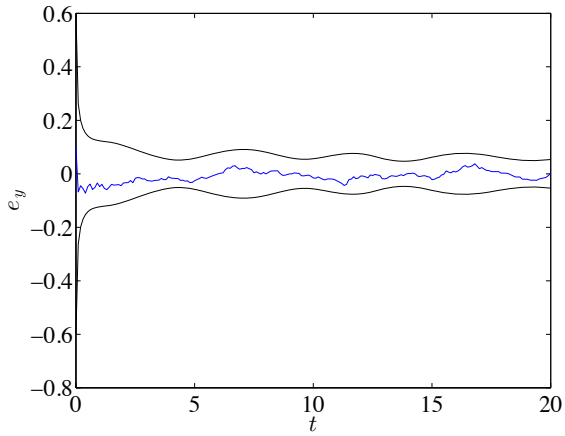
(a) Position



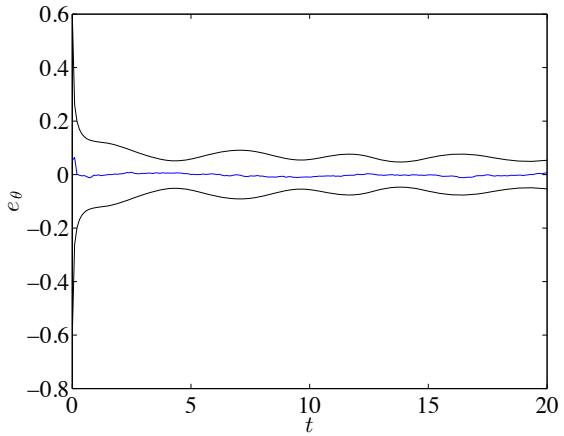
(b) Measurement



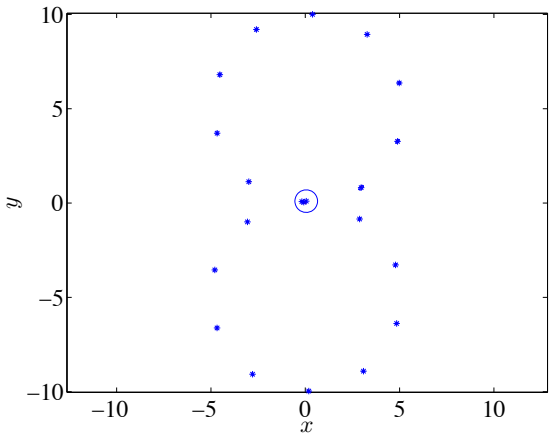
(c) Estimation error e_x



(d) Estimation error e_y



(e) Estimation error e_θ



(f) Gaussian ellipsoid

Figure 1: EKF for Localization

MAE 6292
FINAL EXAM

Harshvardhan Uppaluru

Problem 1. EKF for LOCALIZATION

(a) Equations of motions:

$$x_{k+1} = x_k + h(V + w_{v_k})\cos\theta_k$$
$$y_{k+1} = y_k + h(V + w_{v_k})\sin\theta_k$$
$$\theta_{k+1} = \theta_k + h(u_k + w_{\theta_k})$$

$h \rightarrow$ timestep
$u \rightarrow$ control input
$V \rightarrow$ fixed velocity
$w_k = \begin{bmatrix} w_{v_k} \\ w_{\theta_k} \end{bmatrix} \rightarrow$ process noise

Rewriting the above equations, we have

$$x_{k+1} = x_k + hV\cos\theta_k + hw_{v_k}\cos\theta_k$$
$$y_{k+1} = y_k + hV\sin\theta_k + hw_{v_k}\sin\theta_k$$
$$\theta_{k+1} = \theta_k + hu_k + hw_{\theta_k}$$

writing them in a matrix form, we have

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} x_k + hV\cos\theta_k \\ y_k + hV\sin\theta_k \\ \theta_k \end{bmatrix}}_{f(x_k, k)} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}}_{B_k} u_k + \underbrace{\begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & 1 \end{bmatrix}}_{G_k} \begin{bmatrix} w_{v_k} \\ w_{\theta_k} \end{bmatrix}$$

Therefore we have

$$X_{k+1} = f(x_k, k) + B_k u_k + G_k w_k$$

* Let nominal reference state be denoted as x_k^R. So we have

$$X_{k+1}^R = f(x_k^R, k) \quad \text{with initial condition } x_0^R.$$

* Defining the error between X_{k+1} and x_{k+1}^R, we get

$$\delta x_{k+1} = X_{k+1} - x_{k+1}^R$$

$$\delta x_{k+1} = f(X_k, k) + B_k u_k + G_k w_k - x_{k+1}^R$$

making a Taylor series expansion of $f(X_k, k)$ about the value x_k^R and dropping all terms except constants and linear terms, we get

$$f(X_k, k) = f(x_k^R, k) + \left.\frac{\partial f(x,k)}{\partial x}\right|_{x=x_k^R} (X_k - x_k^R)$$

let $$A_k = \left.\frac{\partial f(x,k)}{\partial x}\right|_{x=x_k^R}$$

Substituting $f(X_k, k)$ into δx_{k+1}, we have

$$\delta x_{k+1} = \cancel{f(x_k^R, k)} + A_k(X_k - x_k^R) + B_k u_k + G_k w_k \cdot \cancel{- f(x_k^R, k)}$$

$$\delta x_{k+1} = A_k \delta x_k + B_k u_k + G_k w_k$$

where

$$B_k = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix} \quad G_k = \begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \end{bmatrix} \quad \delta x_k = (X_k - x_k^R) \quad \text{and} \ldots .$$

where A_k is a $n \times n$ matrix

$$A_k = \left.\frac{\partial f(x,k)}{\partial x}\right|_{x=x_k^R} = \left[\begin{array}{ccc} \frac{\partial f_1(x,k)}{\partial x_1} & \frac{\partial f_1(x,k)}{\partial x_2} & \frac{\partial f_1(x,k)}{\partial x_3} \\ \frac{\partial f_2(x,k)}{\partial x_1} & \frac{\partial f_2(x,k)}{\partial x_2} & \frac{\partial f_2(x,k)}{\partial x_3} \\ \frac{\partial f_3(x,k)}{\partial x_1} & \frac{\partial f_3(x,k)}{\partial x_2} & \frac{\partial f_3(x,k)}{\partial x_3} \end{array}\right]_{x=x_k^R}$$

where

$$f_1(x,k) = x_k + hV\cos\theta_k$$

$$f_2(x,k) = y_k + hV\sin\theta_k$$

$$f_3(x,k) = \theta_k$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Finally, we have

$$A_k = \begin{bmatrix} 1 & 0 & -hV\sin\theta_k \\ 0 & 1 & hV\cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\delta x_{k+1} = A_k \delta x_k + B_k u_k + G_k \omega_k}$$

(b) We have $z^i = \left[z_r^i , z_\theta^i \right]^T$

$$z^i = \begin{bmatrix} z_r^i \\ z_\theta^i \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta x_i^2 + \Delta y_i^2} + v_r \\ \tan^{-1}\left(\frac{\Delta y_i^{body}}{\Delta x_i^{body}} \right) + v_\theta \end{bmatrix}$$

$$z^i = h^i(x_k, k) + v_k$$

Expanding $h^i(x_k, k)$ in a Taylor series about nominal state x_k^R.

$$h^i(x_k, k) \approx h(x_k^R, k) + \left. \frac{\partial h(x,k)}{\partial x} \right|_{x = x_k^R} (x_k - x_k^R)$$

Define $H^i = \left. \frac{\partial h(x,k)}{\partial x} \right|_{x = x_k^R}$

Let nominal measurements be $z_k^R = h(x_k^R, k) + v_k$

$$\delta z^i = z^i - z_k^R$$

$$= h(x_k^R, k) + H^i (x_k - x_k^R) - h(x_k^R, k)$$

$$\delta z^i = H^i (x_k - x_k^R)$$

$$\delta z^i = H^i \delta x$$

where $H^i = \frac{\partial h(x,k)}{\partial x}$

$$H^i = \begin{bmatrix} \frac{\partial h_1(x,k)}{\partial x_1} & \frac{\partial h_1(x,k)}{\partial x_2} & \frac{\partial h_1(x,k)}{\partial x_3} \\ \frac{\partial h_2(x,k)}{\partial x_1} & \frac{\partial h_2(x,k)}{\partial x_2} & \frac{\partial h_2(x,k)}{\partial x_3} \end{bmatrix}$$

where

$$h_1(x,K) = \sqrt{\Delta x_i^2 + \Delta y_i^2} \rightarrow \sqrt{(r_x^i - x)^2 + (r_y^i - y)^2}$$

$$h_2(x,K) = \tan^{-1}\left(\frac{\Delta y_i^{body}}{\Delta x_i^{body}}\right)$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Therefore we have

$$H^i = \begin{bmatrix} \frac{-\Delta x_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}} & \frac{-\Delta y_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}} & 0 \\ \frac{\Delta y_i}{\Delta x_i^2 + \Delta y_i^2} & -\frac{\Delta x_i}{\Delta x_i^2 + \Delta y_i^2} & -1 \end{bmatrix}$$

Hence we have

$$\boxed{\delta z^i = H^i \delta x}$$

```
clc;
clear all;
close all;
```

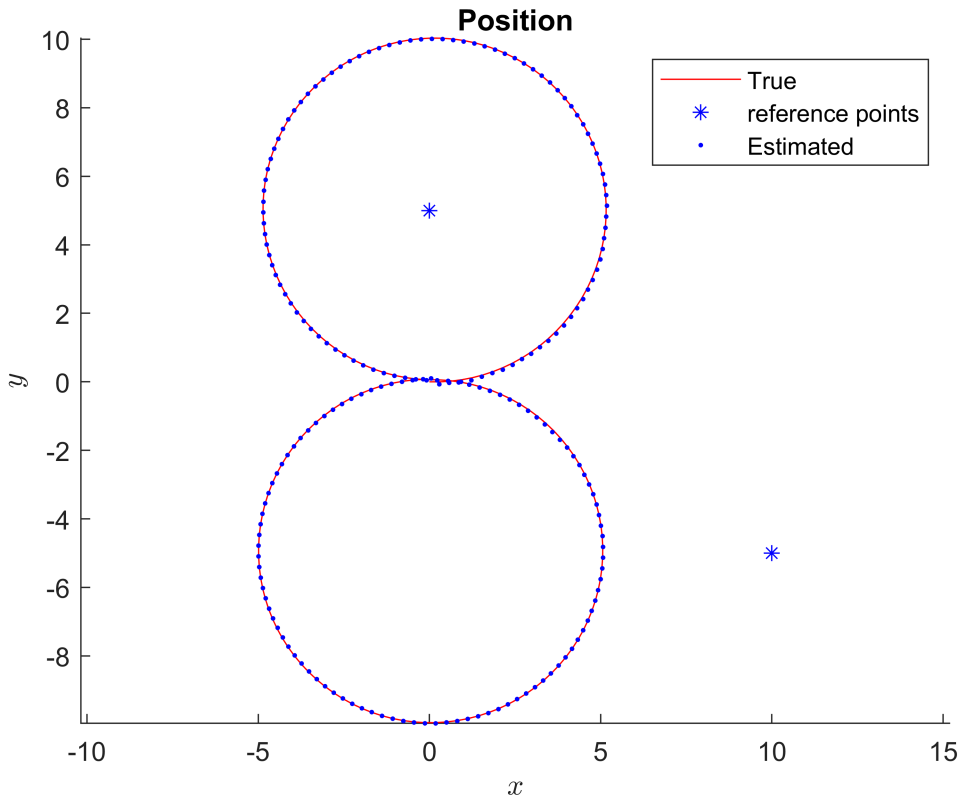
```
N=201;
t=linspace(0,20,N);
h=t(2)-t(1);
sigma_wV=1e-2;
sigma_wt=1e-2;
sigma_vr=1e-1;
sigma_vt=3*pi/180;

Nref=2;
ref=[0 10; 5 -5];

Q=diag([sigma_wV^2 sigma_wt^2]);
R=diag([sigma_vr^2 sigma_vt^2 sigma_vr^2 sigma_vt^2]);

X0=[0.05 0.1 3*pi/180]';
P0=diag([0.2^2 0.2^2 (5*pi/180)^2]);

load prob1_wv;

X_true=zeros(3,N);
X_true(:,1)=[0 0 0]';

V=pi;

% True Trajectory
for k=1:N-1
    if k <= (N-1)/2
        u(k)=pi/5;
    else
        u(k)=-pi/5;
    end
    theta_k=X_true(3,k);
    X_true(:,k+1)=X_true(:,k)...
        +[h*(V+w_V(k))*cos(theta_k);h*(V+w_V(k))*sin(theta_k);h*(u(k)+w_t(k))];
end

% Measurement
for k=1:N
    x=X_true(1,k);
    y=X_true(2,k);
    theta=X_true(3,k);

    for j=1:Nref
        rx=ref(1,j);
        ry=ref(2,j);

        delx=rx-x;
        dely=ry-y;
```

```
        dis=sqrt(delx^2+dely^2);

        delP=[delx; dely];
        delP_body=[cos(theta) sin(theta);
            -sin(theta) cos(theta)]*delP;
        angle=atan2(delP_body(2),delP_body(1));

        z(2*j-1:2*j,k)=[dis; angle];
    end

    z(:,k)=z(:,k)+v(:,k);
end
```

```
% EKF

X_bar=zeros(3,N);
P=zeros(3,3,N);
X_bar(:,1)=X0;
P(:,:,1)=P0;
Z_bar=zeros(2*Nref,N);

for l=1:N-1

    % Prediction
    prev_theta = X_bar(3,l);
    prev_x = X_bar(1,l);
    prev_y = X_bar(2,l);
    curr_x = prev_x + h*V*cos(prev_theta);
    curr_y = prev_y + h*V*sin(prev_theta);
    curr_theta = prev_theta + h*u(l);
    Ak = [1 0 -h*V*sin(prev_theta);
        0 1 h*V*cos(prev_theta);
        0 0 1];
    Bk = [0; 0; h];
    Gk = [h*cos(prev_theta) 0; h*sin(prev_theta) 0; 0 h];
    X_bar_kp = [curr_x curr_y curr_theta]';
    P_kp = Ak*P(:,:,l)*Ak' + Gk*Q*Gk';

    % Correction:
    for n=1:Nref
        refx = ref(1,n);
        refy = ref(2,n);
        delx = refx - curr_x;
        dely = refy - curr_y;
        dist = delx^2 + dely^2;
        % 2x1 matrix
        delP = [delx;dely];
        delP_body = [cos(curr_theta) sin(curr_theta);
            -sin(curr_theta) cos(curr_theta)]*delP;
        z_theta = atan2(delP_body(2), delP_body(1));
        % For each measurement, we have a 2x1 matrix
        Z_bar(2*n-1:2*n,l) = [sqrt(dist);z_theta];
        % 2x3 matrix
```

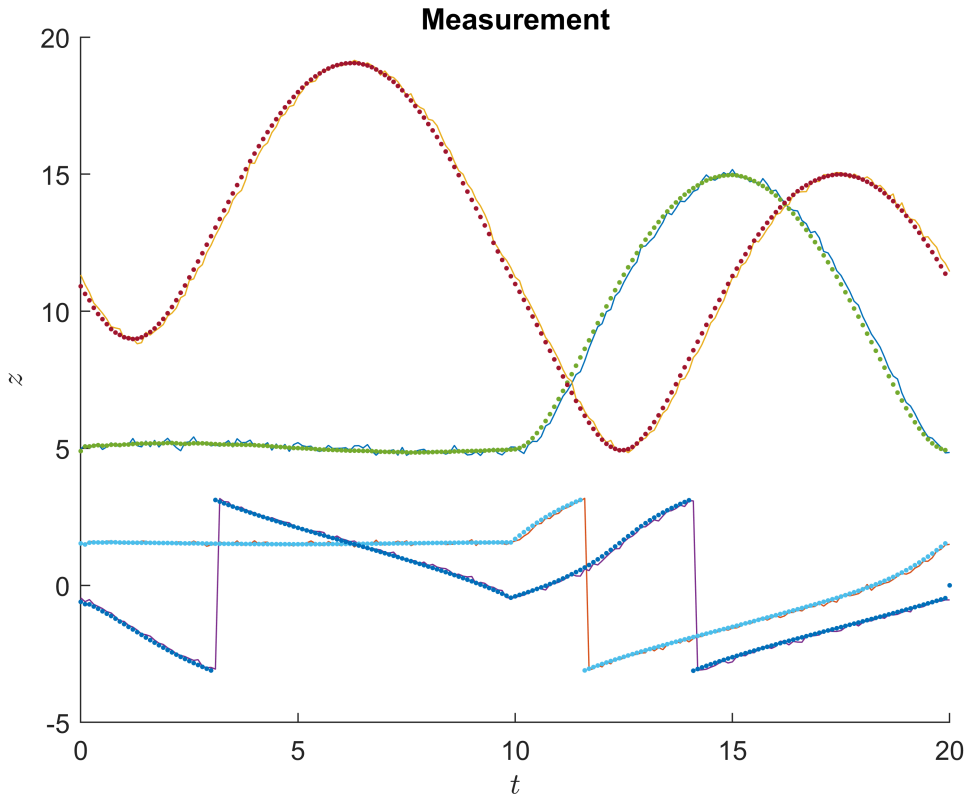
```
        H = [-delx/sqrt(dist) -dely/sqrt(dist) 0;
            dely/dist -delx/dist -1];
        S = H*P_kp*H' + R(2*n-1:2*n,2*n-1:2*n);
        K = P_kp * H'/S;
        X_bar_kp = X_bar_kp + K*(z(2*n-1:2*n,l+1) - Z_bar(2*n-1:2*n,l));
        P_kp = (eye(3) - K*H)*P_kp;
    end
    P(:,:,l+1) = P_kp;
    X_bar(:,l+1) = X_bar_kp;
end
```

```
% Plot 1:
figure;
title('Position'); hold on
plot(X_true(1,:),X_true(2,:),'r');
plot(ref(1,:),ref(2,:),'b*');
plot(X_bar(1,:),X_bar(2,:),'b.')
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');
axis equal;
legend('True', 'reference points', 'Estimated')
hold off;
```

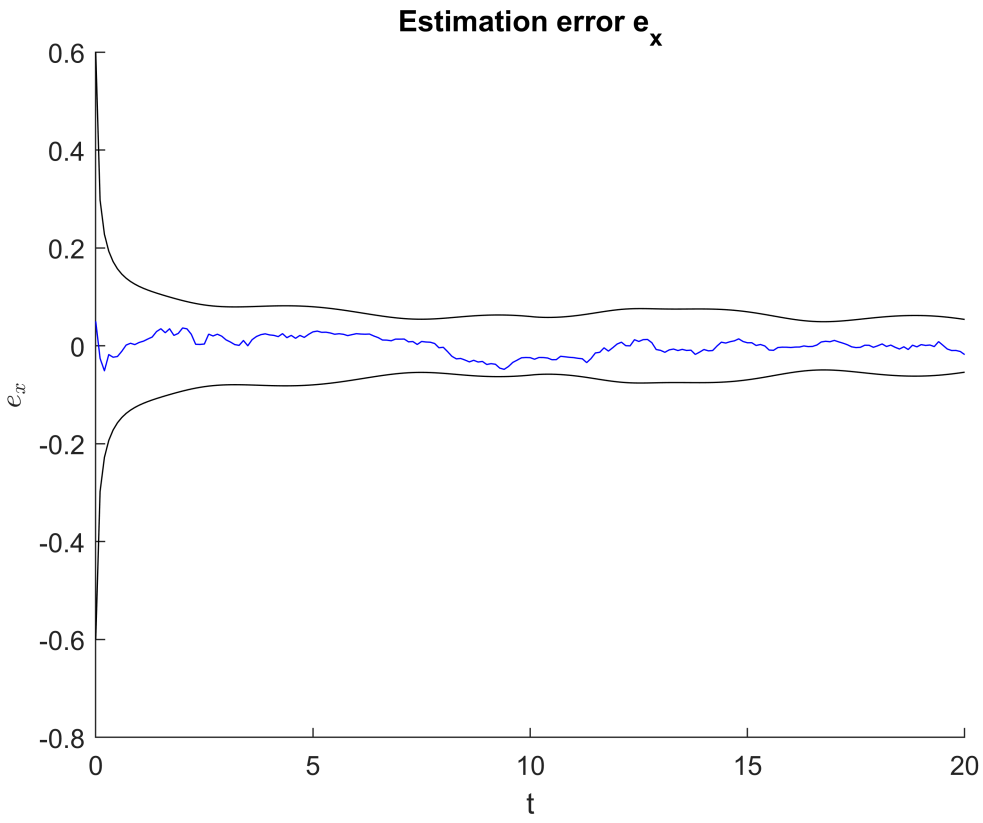



```
% Plot 2:
figure;
title("Measurement"); hold on;
plot(t,z);
```

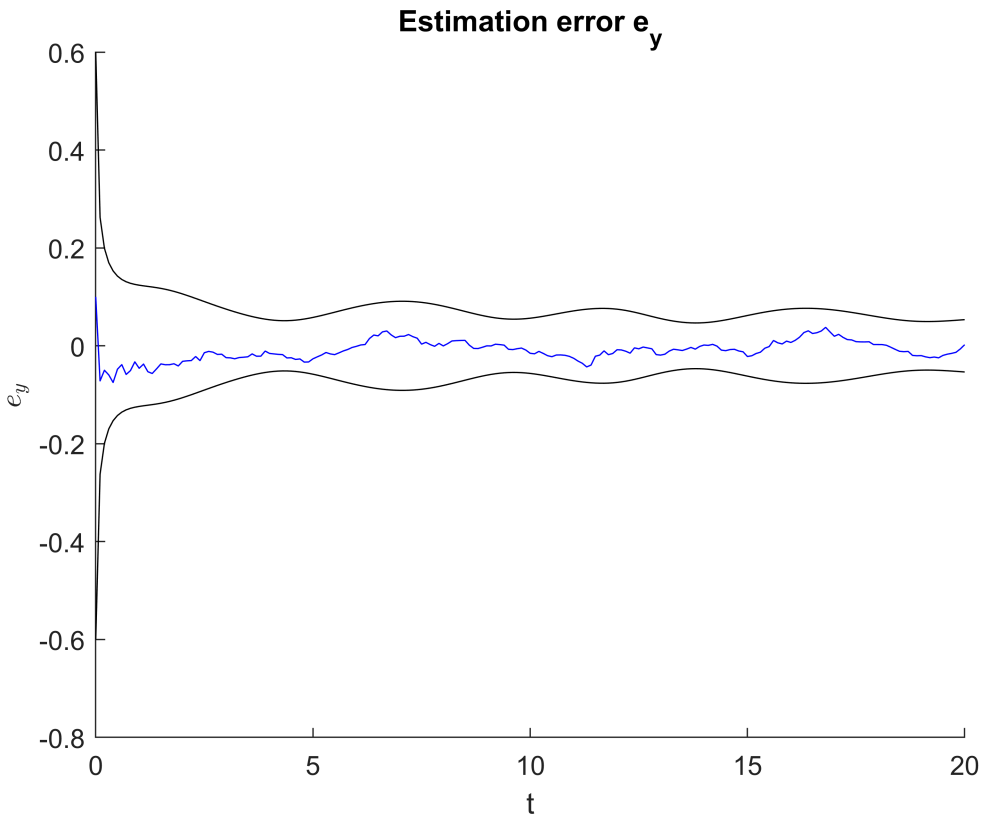
```
plot(t,Z_bar, '.');
xlabel('$t$','interpreter','latex');
ylabel('$z$','interpreter','latex');
%legend('True', 'Estimated')
hold off
```



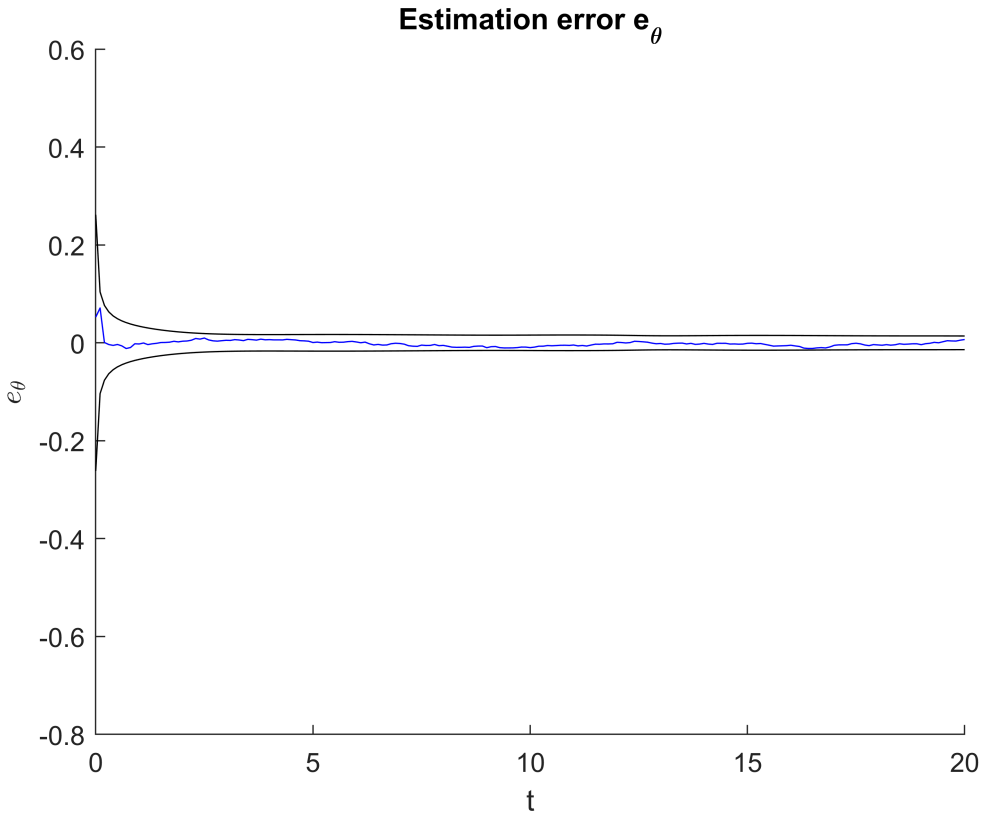

```
% Error Plotting
for m = 1:N
    sigma_x(m) = sqrt(P(1,1,m));
    sigma_y(m) = sqrt(P(2,2,m));
    sigma_theta(m) = sqrt(P(3,3,m));
end

% Plot 3:
diffX = X_bar(1,:) - X_true(1,:);
figure
title('Estimation error e_x'); hold on;
plot(t,diffX, 'b',t,3*sigma_x,'k',t,-3*sigma_x,'k');
xlabel('t');
ylabel('$e_x$','interpreter','latex');
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```



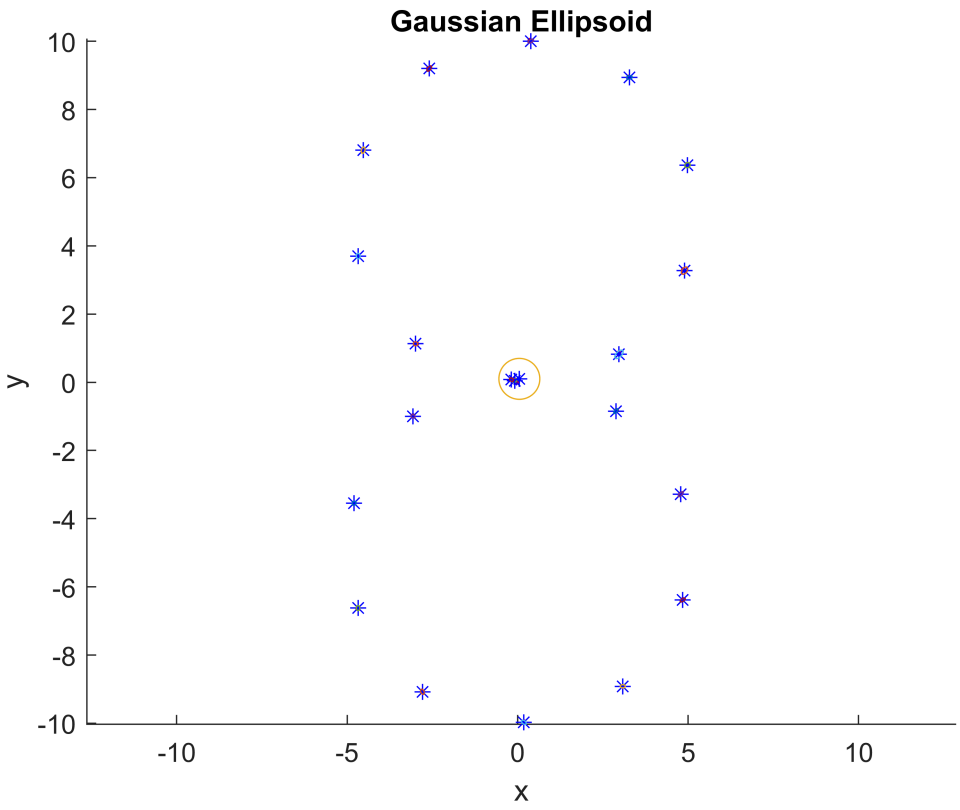

```
% Plot 4:
diffY = X_bar(2,:) - X_true(2,:);
figure
title('Estimation error e_y'); hold on;
plot(t,diffY, 'b',t,3*sigma_y,'k',t,-3*sigma_y,'k');
xlabel('t');
ylabel('$e_y$','interpreter','latex');
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```




```
% Plot 5:
diffTheta = X_bar(3,:) - X_true(3,:);
figure
title('Estimation error e_{\theta}'); hold on;
plot(t,diffTheta, 'b',t,3*sigma_theta,'k',t,-3*sigma_theta,'k');
xlabel('t');
ylabel('$e_\theta$','interpreter','latex');
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```



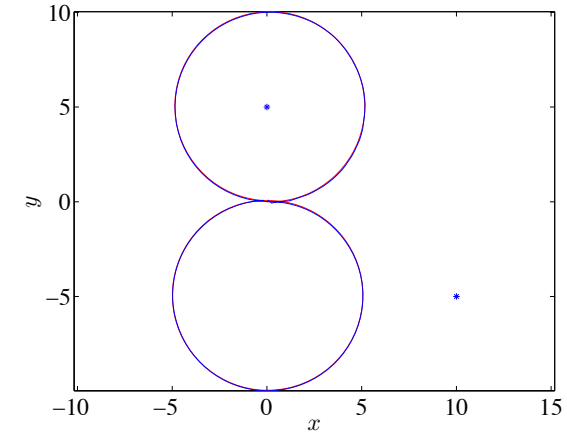

```
% Plot 6:
figure
title('Gaussian Ellipsoid'); hold on;
for a=1:10:N
    plot(X_true(1,a), X_true(2,a),'r.');hold on;
    plot(X_bar(1,a), X_bar(2,a),'b*');
    plot_gaussian_ellipsoid(X_bar([1,2],a),P([1 2],[1,2],a),3);
end
axis equal;
xlabel('x');
ylabel('y');
```


Gaussian Ellipsoid
10
8
6
4
2
0
-2
-4
-6
-8
-10
y
-10
-5
0
5
10
x

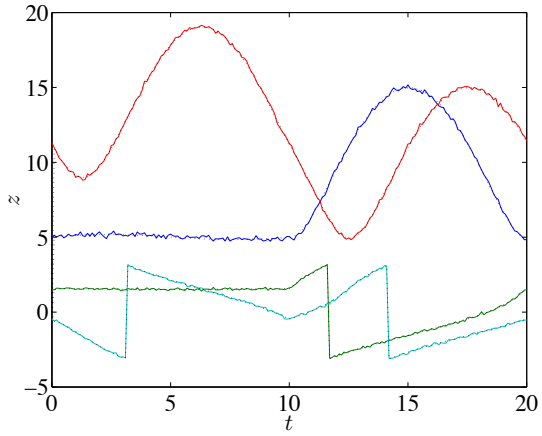
Problem 2 (UKF for Localization) Consider the robot model discussed at Problem 1.

(a) All of the parameters for the robot model, control inputs, measurements, reference points, and initial guess are specified at `prob2.m`. Write a Matlab file for UKF to estimate the state $\mathbf{x}_k$, and generate 6 plots given at the next page.

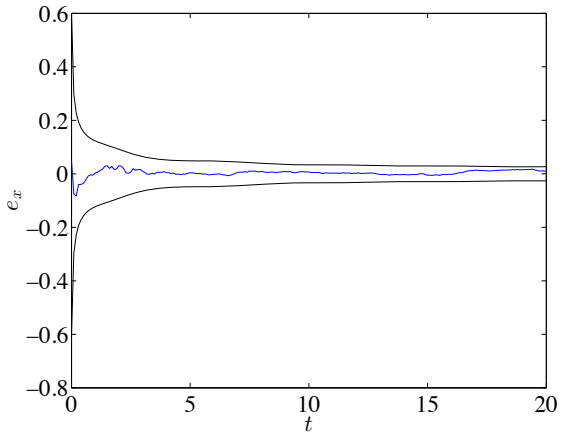
(b) Compare the performance of UKF with EKF for this robot model.



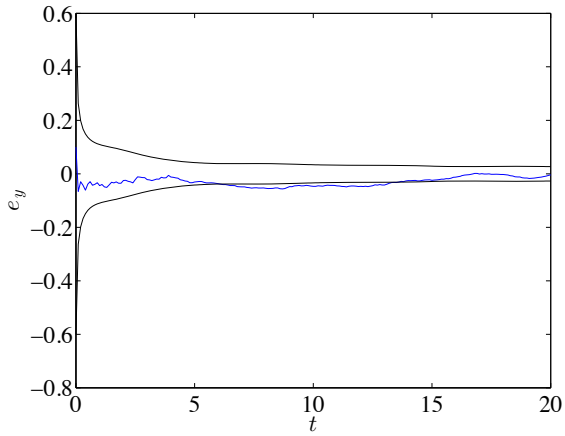
(a) Position



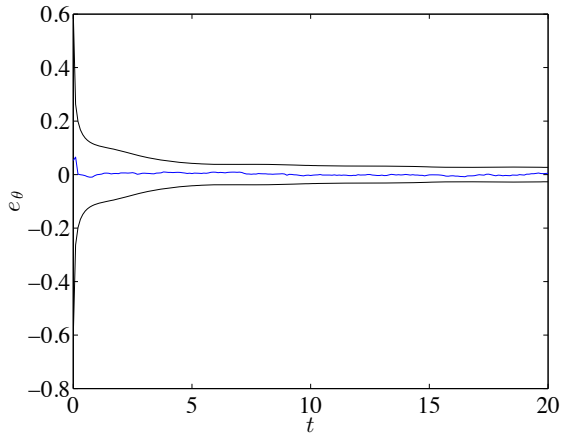
(b) Measurement



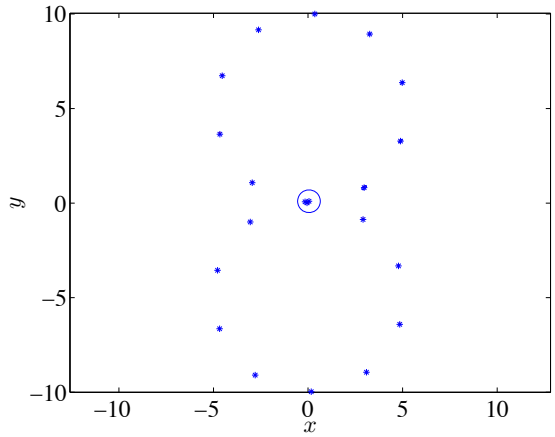
(c) Estimation error e_x



(d) Estimation error e_y



(e) Estimation error e_θ



(f) Gaussian ellipsoid

Figure 2: UKF for Localization

```
clc;
clear all;
close all;
```

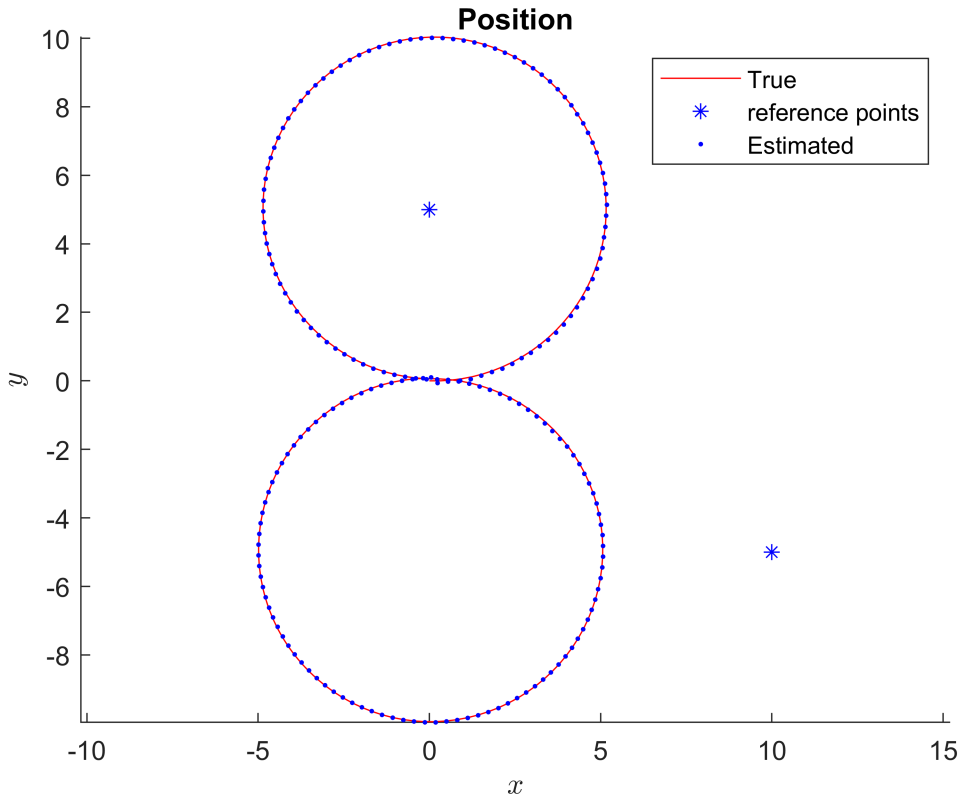
```
N=201;
t=linspace(0,20,N);
h=t(2)-t(1);
sigma_wV=1e-2;
sigma_wt=1e-2;
sigma_vr=1e-1;
sigma_vt=3*pi/180;

Nref=2;
ref=[0 10; 5 -5];

Q=diag([sigma_wV^2 sigma_wt^2]);
R=diag([sigma_vr^2 sigma_vt^2 sigma_vr^2 sigma_vt^2]);

X0=[0.05 0.1 3*pi/180]';
P0=diag([0.2^2 0.2^2 (5*pi/180)^2]);

load prob1_wv;

X_true=zeros(3,N);
X_true(:,1)=[0 0 0]';

V=pi;

% True Trajectory
for k=1:N-1
    if k <= (N-1)/2
        u(k)=pi/5;
    else
        u(k)=-pi/5;
    end
    theta_k=X_true(3,k);
    X_true(:,k+1)=X_true(:,k)...
        +[h*(V+w_V(k))*cos(theta_k);h*(V+w_V(k))*sin(theta_k);h*(u(k)+w_t(k))];
end

% Measurement
for k=1:N
    x=X_true(1,k);
    y=X_true(2,k);
    theta=X_true(3,k);

    for j=1:Nref
        rx=ref(1,j);
        ry=ref(2,j);

        delx=rx-x;
        dely=ry-y;
```

```
        dis=sqrt(delx^2+dely^2);

        delP=[delx; dely];
        delP_body=[cos(theta) sin(theta);
            -sin(theta) cos(theta)]*delP;
        angle=atan2(delP_body(2),delP_body(1));

        z(2*j-1:2*j,k)=[dis; angle];
    end

    z(:,k)=z(:,k)+v(:,k);
end
```

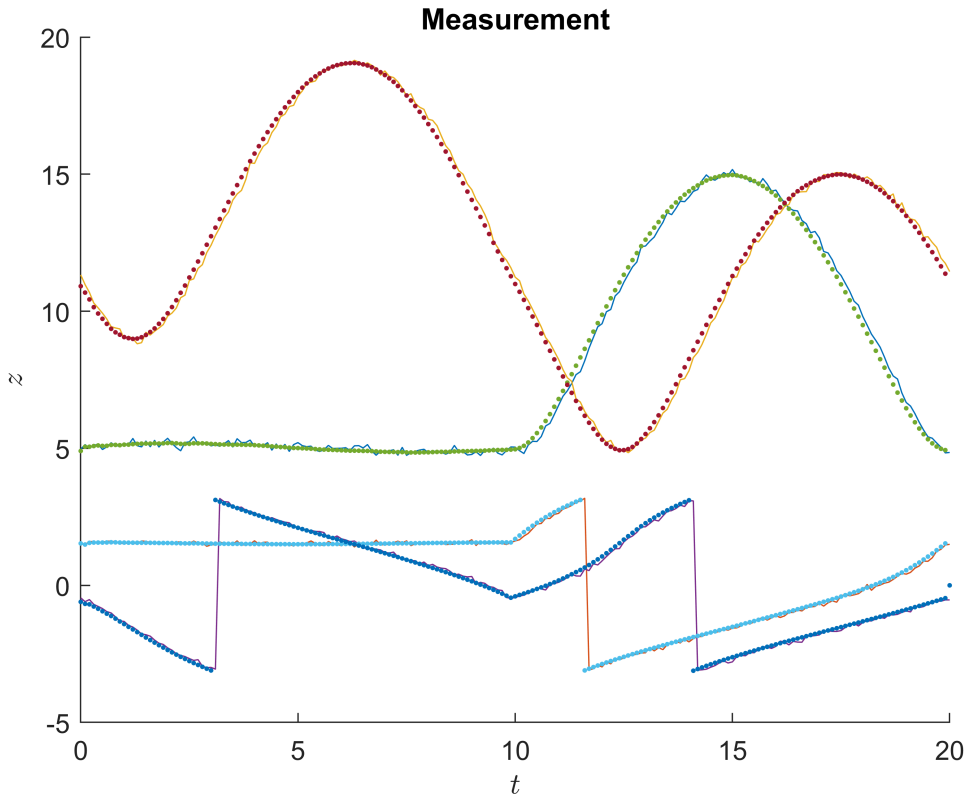
```
% UKF
kappa=50;
alpha=0.25;
beta=2;
n=3;


X_bar=zeros(3,N);
P=zeros(3,3,N);
X_bar(:,1)=X0;
P(:,:,1)=P0;
Z_bar=zeros(2*Nref,N);
for k=1:N-1
    % prediction
    % 1: choose sigma points
    % X_k_sigma = 3x7 matrix
    [X_k_sigma, W_m, W_v]=UT_sigma(X_bar(:,k), P(:,:,k),kappa,alpha,beta);
    % 2: Pass sigma points through motion model and compute Gaussian statistics
    X_kp_sigma = zeros(n,2*n+1);
    for i=1:2*n+1
        x = X_k_sigma(1,i);
        y = X_k_sigma(2,i);
        theta = X_k_sigma(3,i);
        curr_x = x + h*V*cos(theta);
        curr_y = y + h*V*sin(theta);
        curr_theta = theta + h*u(k);
        Ak = [1 0 -h*V*sin(theta);
            0 1 h*V*cos(theta);
            0 0 1];
        Bk = [0; 0; h];
        X_kp_sigma(:,i) = [curr_x; curr_y; curr_theta];
    end
    % 3: Recover mean and variance
    [X_bar_kp, P_kp] = UT_recover(X_kp_sigma, W_m, W_v);
    % 4: Add process noise
    Gk = [h*cos(theta) 0; h*sin(theta) 0; 0 h];
    P_kp = P_kp + Gk*Q*Gk';
    % Correction
    % 1: Get sigma points
    [X_sigma, W_m, W_v] = UT_sigma(X_bar_kp, P_kp, kappa, alpha, beta);
```

```
    % 2: Generate measurement for each sigma point
    Z_sigma = zeros(4,2*n+1);
    for i = 1:2*n+1
        curr_x = X_sigma(1,i);
        curr_y = X_sigma(2,i);
        curr_theta = X_sigma(3,i);
        for j = 1:Nref
            refx = ref(1,j);
            refy = ref(2,j);
            delX = refx - curr_x;
            delY = refy - curr_y;
            dist = sqrt(delX^2 + delY^2);
            delP = [delX;delY];
            delP_body = [cos(curr_theta) sin(curr_theta);
                -sin(curr_theta) cos(curr_theta)]*delP;
            z_theta = atan2(delP_body(2), delP_body(1));
            % Finally gives out a 4x7 matrix
            Z_sigma(2*j-1:2*j,i) = [dist; z_theta];
        end
    end
    % 3: recover mean and variance
    [Z_bar(:,k), P_z] = UT_recover(Z_sigma, W_m, W_v);
    P_z = P_z+R;
    P_xz = zeros(n,4);
    for i=1:2*n+1
        P_xz = P_xz + W_v(i) * (X_sigma(:,i) - X_bar_kp)*(Z_sigma(:,i)-Z_bar(:,k))';
    end
    % 4: Update
    K = P_xz * inv(P_z);
    X_bar_kp = X_bar_kp + K*(z(:,k+1) - Z_bar(:,k));
    P_kp = P_kp - K*P_z*K';
    P(:,:,k+1) = P_kp;
    X_bar(:,k+1) = X_bar_kp;
end
```

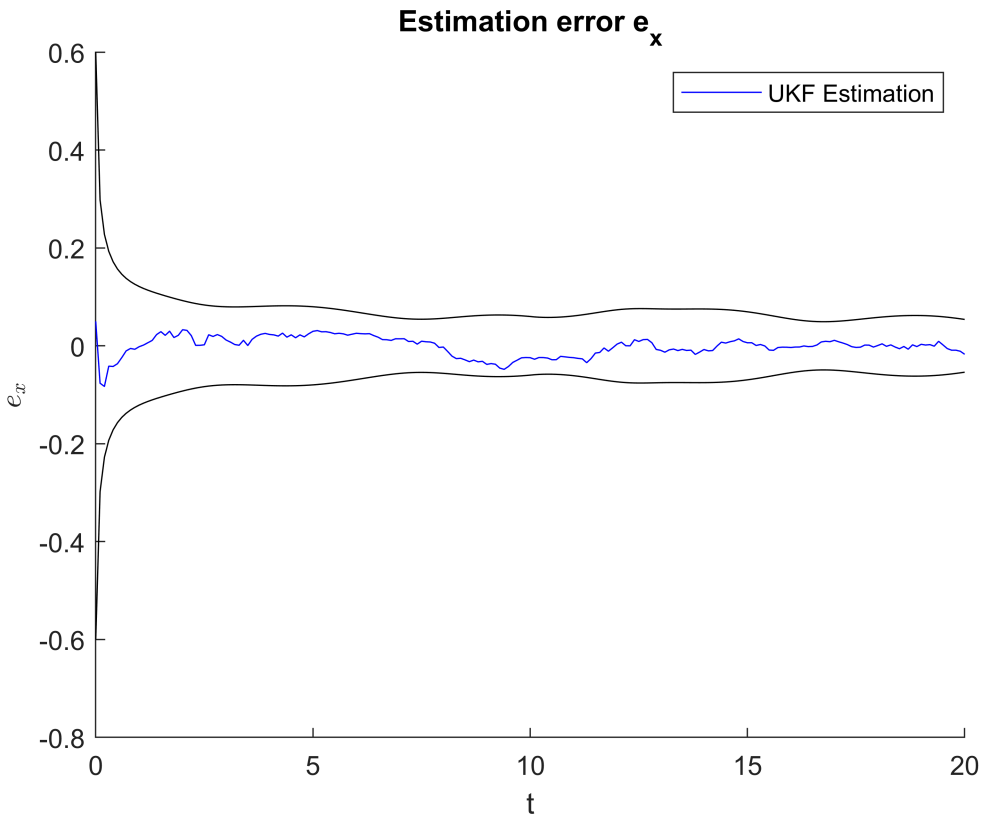
```
% Plot 1:
figure(1);
title('Position'); hold on;
plot(X_true(1,:),X_true(2,:),'r');
plot(ref(1,:),ref(2,:),'b*');
plot(X_bar(1,:),X_bar(2,:),'b.')
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');
axis equal;
legend('True', 'reference points', 'Estimated');
hold off;
```



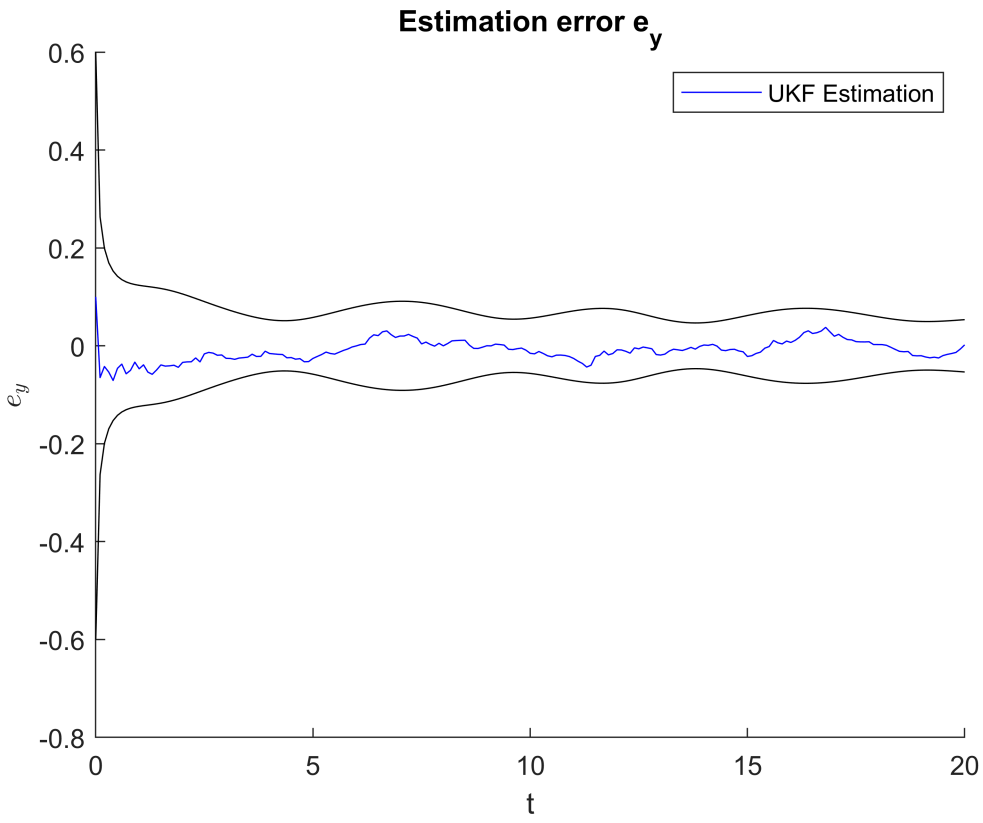

```
% Plot 2:
figure(2);
title("Measurement"); hold on;
plot(t,z);
plot(t,Z_bar,'.');
xlabel('$t$','interpreter','latex');
ylabel('$z$','interpreter','latex');
hold off;
```



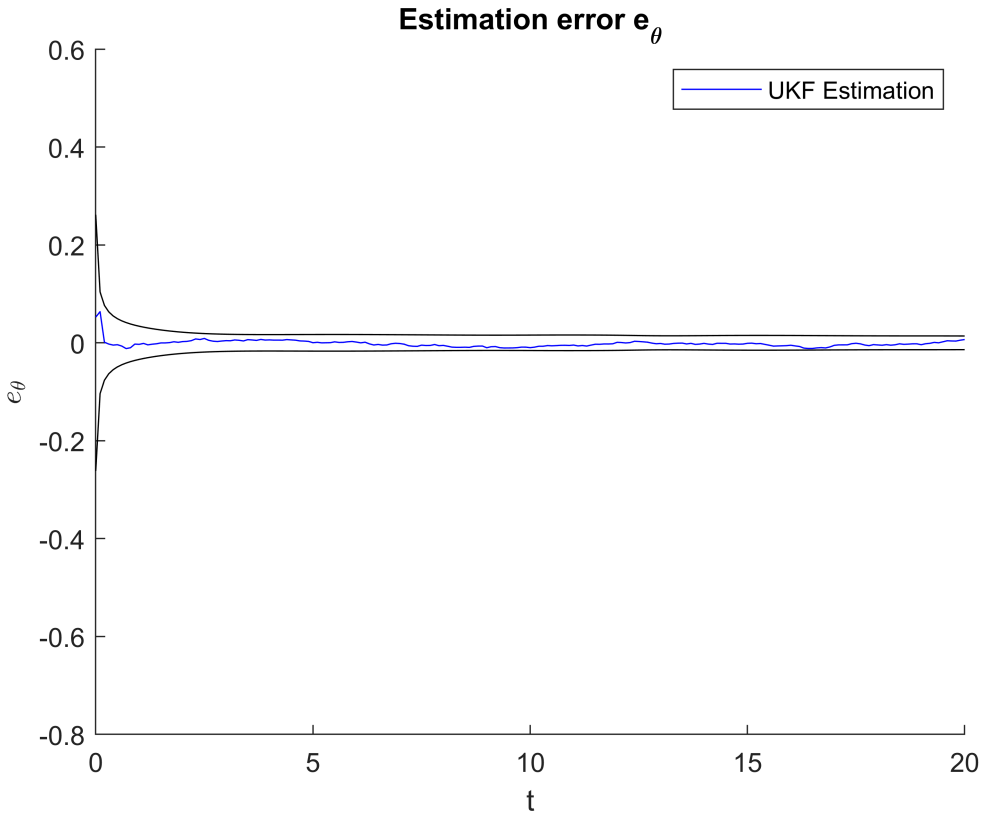

```
% Error Plotting
for m = 1:N
    sigma_x(m) = sqrt(P(1,1,m));
    sigma_y(m) = sqrt(P(2,2,m));
    sigma_theta(m) = sqrt(P(3,3,m));
end

% Plot 3:
diffX = X_bar(1,:) - X_true(1,:);
figure(3);
title('Estimation error e_x'); hold on;
plot(t,diffX, 'b',t,3*sigma_x,'k',t,-3*sigma_x,'k');
xlabel('t');
ylabel('$e_x$','interpreter','latex');
legend('UKF Estimation')
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```



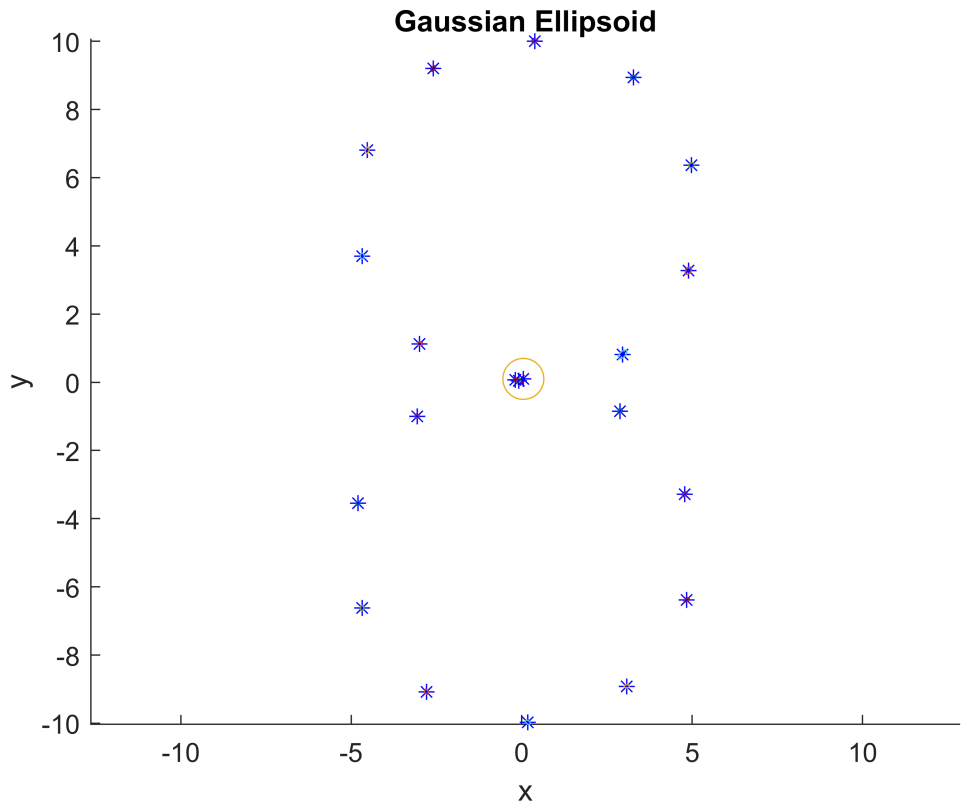

```
% Plot 4:
diffY = X_bar(2,:) - X_true(2,:);
figure(4);
title('Estimation error e_y'); hold on;
plot(t,diffY, 'b',t,3*sigma_y,'k',t,-3*sigma_y,'k');
xlabel('t');
ylabel('$e_y$','interpreter','latex');
legend('UKF Estimation')
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```




```
% Plot 5:
diffTheta = X_bar(3,:) - X_true(3,:);
figure(5);
title('Estimation error e_{\theta}'); hold on;
plot(t,diffTheta, 'b',t,3*sigma_theta,'k',t,-3*sigma_theta,'k');
xlabel('t');
ylabel('$e_\theta$','interpreter','latex');
legend('UKF Estimation')
xlim([0 20.0])
ylim([-0.8 0.60])
hold off;
```



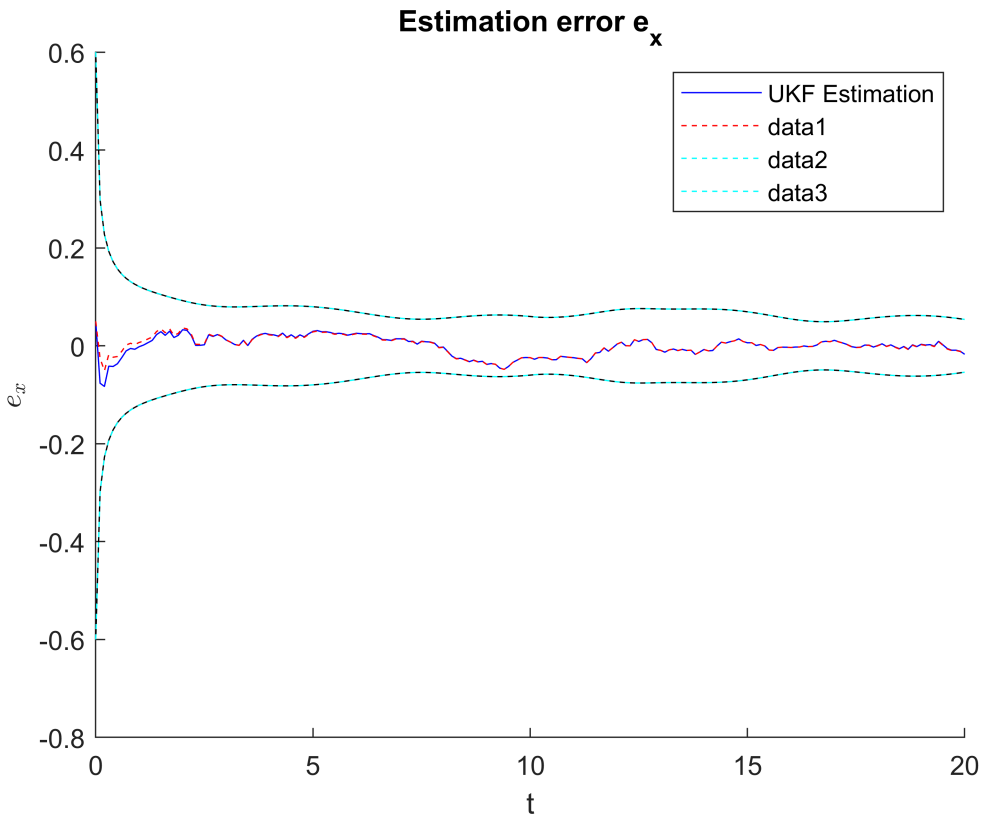

```
% Plot 6:
figure(6);
title('Gaussian Ellipsoid'); hold on;
for a=1:10:N
    plot(X_true(1,a), X_true(2,a),'r.');hold on;
    plot(X_bar(1,a), X_bar(2,a),'b*');
    plot_gaussian_ellipsoid(X_bar([1,2],a),P([1 2],[1,2],a),3);
end
axis equal;
xlabel('x');
ylabel('y');
hold off;
```



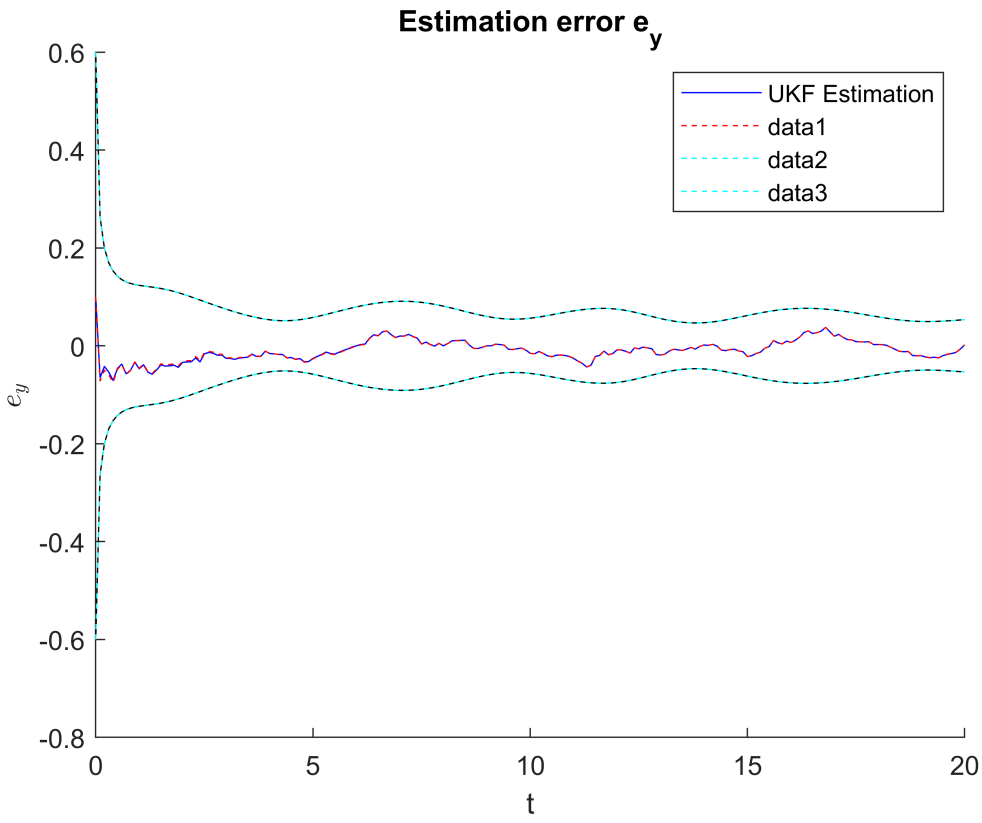

```
% Comparison with EKF
X_bar=zeros(3,N);
P=zeros(3,3,N);
X_bar(:,1)=X0;
P(:,:,1)=P0;
Z_bar=zeros(2*Nref,N);

for l=1:N-1

    % Prediction
    prev_theta = X_bar(3,l);
    prev_x = X_bar(1,l);
    prev_y = X_bar(2,l);
    curr_x = prev_x + h*V*cos(prev_theta);
    curr_y = prev_y + h*V*sin(prev_theta);
    curr_theta = prev_theta + h*u(l);
    Ak = [1 0 -h*V*sin(prev_theta);
        0 1 h*V*cos(prev_theta);
        0 0 1];
    Bk = [0; 0; h];
    Gk = [h*cos(prev_theta) 0; h*sin(prev_theta) 0; 0 h];
    X_bar_kp = [curr_x curr_y curr_theta]';
    P_kp = Ak*P(:,:,l)*Ak' + Gk*Q*Gk';

    % Correction:
    for n=1:Nref
        refx = ref(1,n);
```

```
        refy = ref(2,n);
        delx = refx - curr_x;
        dely = refy - curr_y;
        dist = delx^2 + dely^2;
        % 2x1 matrix
        delP = [delx;dely];
        delP_body = [cos(curr_theta) sin(curr_theta);
            -sin(curr_theta) cos(curr_theta)]*delP;
        z_theta = atan2(delP_body(2), delP_body(1));
        % For each measurement, we have a 2x1 matrix
        Z_bar(2*n-1:2*n,l) = [sqrt(dist);z_theta];
        % 2x3 matrix
        H = [-delx/sqrt(dist) -dely/sqrt(dist) 0;
            dely/dist -delx/dist -1];
        S = H*P_kp*H' + R(2*n-1:2*n,2*n-1:2*n);
        K = P_kp * H'/S;
        X_bar_kp = X_bar_kp + K*(z(2*n-1:2*n,l+1) - Z_bar(2*n-1:2*n,l));
        P_kp = (eye(3) - K*H)*P_kp;
    end
    P(:,:,l+1) = P_kp;
    X_bar(:,l+1) = X_bar_kp;
end
```

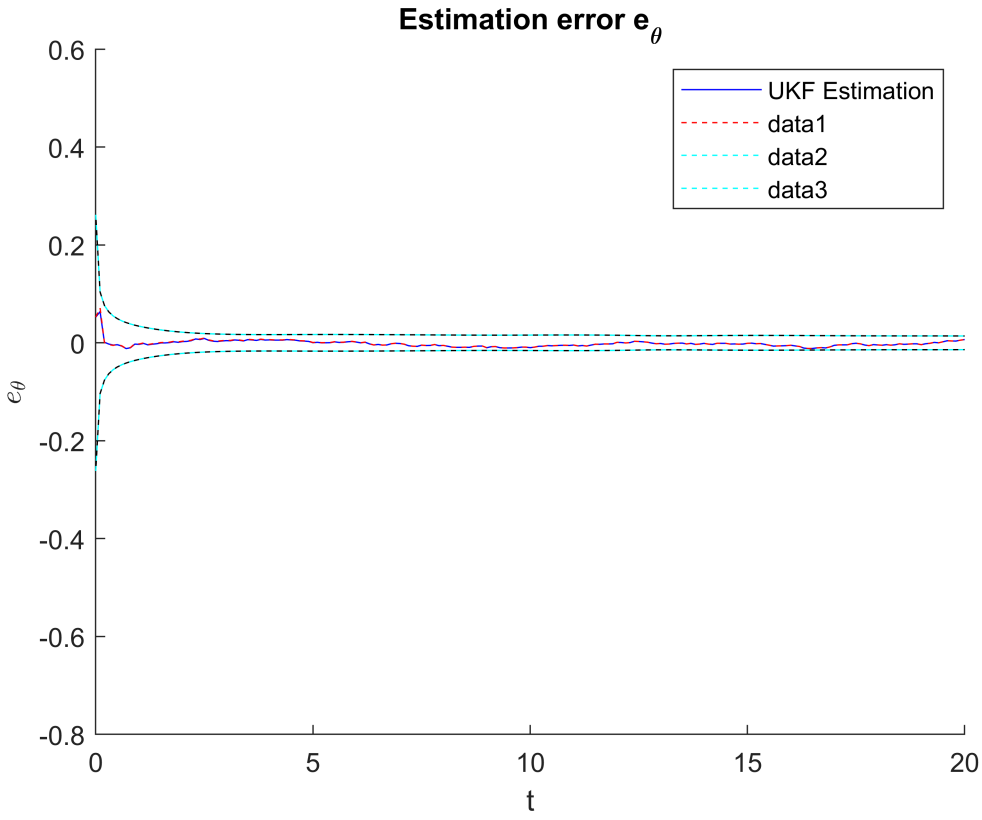
```
% Comparison with EKF plots

% Error Plotting
for m = 1:N
    sigma_x(m) = sqrt(P(1,1,m));
    sigma_y(m) = sqrt(P(2,2,m));
    sigma_theta(m) = sqrt(P(3,3,m));
end

figure(3);hold on;
plot(t,X_bar(1,:)-X_true(1,:),'r--',t,3*sigma_x,'c--',t,-3*sigma_x,'c--');
xlim([0 20.0])
ylim([-0.8 0.60])
```




```
%legend('EKF Estimation', 'EKF 3-{\sigma}', 'EKF 3-{\sigma}')
figure(4);hold on;
plot(t,X_bar(2,:)-X_true(2,:),'r--',t,3*sigma_y,'c--',t,-3*sigma_y,'c--');
xlim([0 20.0])
ylim([-0.8 0.60])
```




```
%legend('EKF Estimation')
figure(5);hold on;
plot(t,X_bar(3,:)-X_true(3,:),'r--',t,3*sigma_theta,'c--',t,-3*sigma_theta,'c--');
xlim([0 20.0])
ylim([-0.8 0.60])
```




```
%legend('EKF Estimation')
```

Problem 3 (EKF for SLAM) Consider the planar robot model discussed at Problem 1. In this question, we assume that the location of the reference points are unknown. The objective is to estimate the position and the orientation of the robot while estimating the location of the reference points as well. This is referred to as *simultaneous localization and mapping* (SLAM).

Since the location of the reference points are unknown. They are included to the state vector. When there are two reference points, the complete state vector for SLAM is given by

$$\mathbf{x} = [x, y, \theta, r_x^1, r_y^1, r_x^2, r_y^2]^T \in \mathbb{R}^7.$$

Assuming that the reference points are stationary, the equations of motion are given by

$$\begin{aligned}
x_{k+1} &= x_k + h(V + w_{v_k})\cos\theta_k, \\
y_{k+1} &= y_k + h(V + w_{v_k})\sin\theta_k, \\
\theta_{k+1} &= \theta_k + h(u_k + w_{\theta_k}), \\
r^1_{x_{k+1}} &= r^1_{x_k}, \\
r^1_{y_{k+1}} &= r^1_{y_k}, \\
r^2_{x_{k+1}} &= r^2_{x_k}, \\
r^2_{y_{k+1}} &= r^2_{y_k}.
\end{aligned}$$

The measurements are identical to Problem 1, i.e., $\mathbf{z} = [z_r^1, z_\theta^1, z_r^2, z_\theta^2]^T \in \mathbb{R}^4$.

(a) Show that the linearized equations of motion are given by

$$\delta\mathbf{x_{k+1}} = A_k\delta\mathbf{x_k} + B_k u_k + G_k w_k,$$

where

$$A_k = \begin{bmatrix} 1 & 0 & -hV\sin\theta_k & 0 & 0 & 0 & 0 \\ 0 & 1 & hV\cos\theta_k & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_k = \begin{bmatrix} 0 \\ 0 \\ h \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad G_k = \begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

(Note: the increase of P_k due to the process noise becomes $G_k Q_k G_k^T$.)

(b) Show that the linearized measurement equation can be written as

$$\delta\mathbf{z} = H\delta\mathbf{x},$$

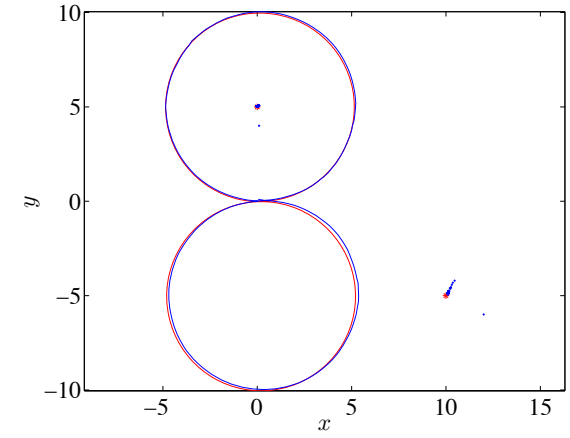
where the matrix $H \in \mathbb{R}^{4\times 7}$ are given by

$$H^i = \begin{bmatrix} \frac{\partial z_r^1}{\partial x} & \frac{\partial z_r^1}{\partial y} & \frac{\partial z_r^1}{\partial \theta} & \frac{\partial z_r^1}{\partial r_x^1} & \frac{\partial z_r^1}{\partial r_y^1} & 0 & 0 \\ \frac{\partial z_\theta^i}{\partial x} & \frac{\partial z_\theta^i}{\partial y} & \frac{\partial z_\theta^i}{\partial \theta} & \frac{\partial z_\theta^1}{\partial r_x^1} & \frac{\partial z_\theta^1}{\partial r_y^1} & 0 & 0 \\ \frac{\partial z_r^2}{\partial x} & \frac{\partial z_r^2}{\partial y} & \frac{\partial z_r^2}{\partial \theta} & 0 & 0 & \frac{\partial z_r^2}{\partial r_x^2} & \frac{\partial z_r^2}{\partial r_y^2} \\ \frac{\partial z_\theta^2}{\partial x} & \frac{\partial z_\theta^2}{\partial y} & \frac{\partial z_\theta^2}{\partial \theta} & 0 & 0 & \frac{\partial z_\theta^2}{\partial r_x^2} & \frac{\partial z_\theta^2}{\partial r_y^2} \end{bmatrix},$$

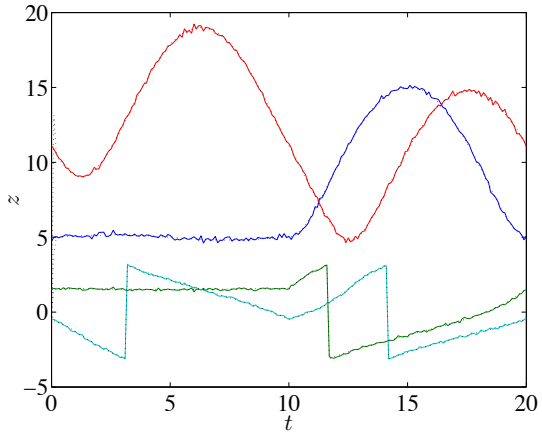
and the derivatives are given by

$$\frac{\partial z_r^i}{\partial r_x^i} = \frac{\Delta x_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}}, \quad \frac{\partial z_r^i}{\partial r_y^i} = \frac{\Delta y_i}{\sqrt{\Delta x_i^2 + \Delta y_i^2}},$$
$$\frac{\partial z_\theta^i}{\partial r_x^i} = -\frac{\Delta y_i}{\Delta x_i^2 + \Delta y_i^2}, \quad \frac{\partial z_\theta^i}{\partial r_y^i} = +\frac{\Delta x_i}{\Delta x_i^2 + \Delta y_i^2}.$$

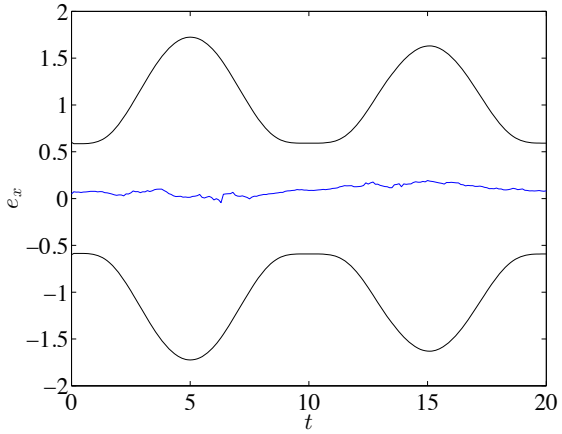
(c) All of the parameters for the robot model, control inputs, measurements, reference points, and initial guess are specified at `prob3.m`. Write a Matlab file for EKF to estimate the state $\mathbf{x}_k$, and generate 6 plots given at the next page.



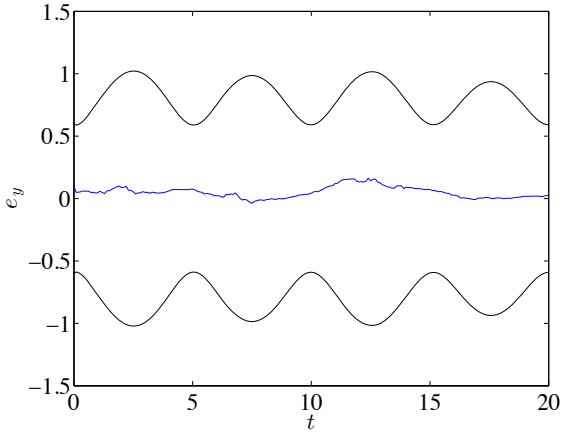
(a) Position



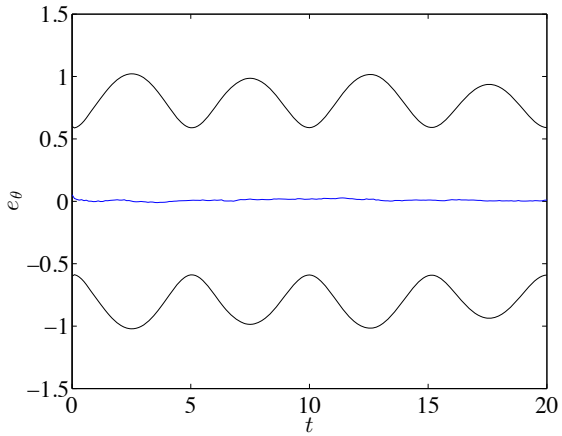
(b) Measurement



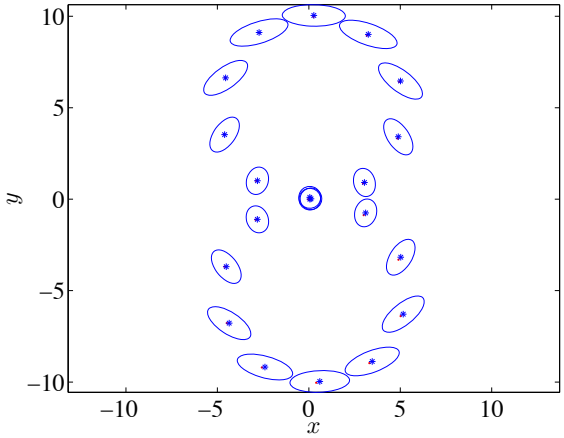
(c) Estimation error e_x



(d) Estimation error e_y



(e) Estimation error e_θ



(f) Gaussian ellipsoid

Figure 3: EKF for SLAM

Problem 3: EKF for SLAM

complete state vector for SLAM: $x = [x, y, \theta, r^1_x, r^1_y, r^2_x, r^2_y]^T \in \mathbb{R}^7$

Assuming reference points are stationary, equations of motion are

$$x_{k+1} = x_k + hV\cos\theta_k + hw_{v_k}\cos\theta_k$$

$$y_{k+1} = y_k + hV\sin\theta_k + hw_{v_k}\sin\theta_k$$

$$\theta_{k+1} = \theta_k + hu_k + hw_{\theta_k}$$

$$r^1_{x_{k+1}} = r^1_{x_k}$$

$$r^1_{y_{k+1}} = r^1_{y_k}$$

$$r^2_{x_{k+1}} = r^2_{x_k}$$

$$r^2_{y_{k+1}} = r^2_{y_k}$$

Measurements:

$$z = [z^1_r, z^1_\theta, z^2_r, z^2_\theta]^T$$

(a) Rewriting the above equations into matrix form, we have

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ r^1_{x_{k+1}} \\ r^1_{y_{k+1}} \\ r^2_{x_{k+1}} \\ r^2_{y_{k+1}} \end{bmatrix} = \underbrace{\begin{bmatrix} x_k + hV\cos\theta_k \\ y_k + hV\sin\theta_k \\ \theta_k \\ r^1_{x_k} \\ r^1_{y_k} \\ r^2_{x_k} \\ r^2_{y_k} \end{bmatrix}}_{f(x_k, k)} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ h \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{B_k} u_k + \underbrace{\begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{G_k} \begin{bmatrix} w_{v_k} \\ w_{\theta_k} \end{bmatrix}$$

We have.

$$\rightarrow \quad X_{k+1} = f(x_k, k) + B_k u_k + G_k w_k$$

* let the nominal state be denoted as x_k^R Therefore, we have

$$x_{k+1}^R = f(x_k^R, k) \quad \text{with initial condition } x_0^R.$$

* The error between x_{k+1} and x_{k+1}^R can be denoted as

$$\delta x_{k+1} = x_{k+1} - x_{k+1}^R$$

$$\delta x_{k+1} = f(x_k, k) + B_k u_k + G_k w_k - x_{k+1}^R$$

writing a Taylor series expansion for $f(x_k, k)$ about the value x_k^R and dropping all terms except constants and linear terms, we get

$$f(x_k, k) = f(x_k^R, k) + \underbrace{\left.\frac{\partial f(x,k)}{\partial x}\right|_{x = x_k^R}}_{A_k} (x_k - x_k^R)$$

Substituting $f(x_k, k)$ into δx_{k+1}, we have

$$\delta x_{k+1} = f(x_k^R, k) + A_k \delta x_k + B_k u_k + G_k w_k - f(x_k^R, k)$$

$$\delta x_{k+1} = A_k \delta x_k + B_k u_k + G_k w_k$$

where

$$B_k = \begin{bmatrix} 0 \\ 0 \\ h \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad G_k = \begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and $A_k = \dfrac{\partial f(x,k)}{\partial x}$

where A_k is a 7×7 matrix

$$A_k = \frac{\partial f(x,k)}{\partial x}$$

$$= \begin{bmatrix} \frac{\partial f_1(x,k)}{\partial x_1} & \frac{\partial f_1(x,k)}{\partial x_2} & \frac{\partial f_1(x,k)}{\partial x_3} & \frac{\partial f_1(x,k)}{\partial x_4} & \frac{\partial f_1(x,k)}{\partial x_5} & \frac{\partial f_1(x,k)}{\partial x_6} & \frac{\partial f_1(x,k)}{\partial x_7} \\ \frac{\partial f_2(x,k)}{\partial x_1} & \cdots & & & & & \\ \vdots & & & & & & \\ \frac{\partial f_7(x,k)}{\partial x_1} & & & & & & \frac{\partial f_7(x,k)}{\partial x_7} \end{bmatrix}$$

where

$$\begin{bmatrix} f_1(x,k) \\ f_2(x,k) \\ f_3(x,k) \\ f_4(x,k) \\ f_5(x,k) \\ f_6(x,k) \\ f_7(x,k) \end{bmatrix} = f(x,k) \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ r^1_x \\ r^1_y \\ r^2_x \\ r^2_y \end{bmatrix}$$

Gives

$$A_k = \begin{bmatrix} 1 & 0 & -hV\sin\theta_k & 0 & 0 & 0 & 0 \\ 0 & 1 & hV\cos\theta_k & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \boxed{\delta x_{k+1} = A_k \delta x_k + B_k u_k + G_k \omega_k}$$

(b) Similarly, we have: $z = [z_r^1, z_\theta^1, z_r^2, z_\theta^2]^T$

where

$$z = h(x_k, k) + v$$

expanding $h(x_k, k)$ in a Taylor series about nominal state x_k^R:

$$h(x_k, k) \approx h(x_k^R, k) + \underbrace{\left.\frac{\partial h(x,k)}{\partial x}\right|_{x=x_k^R}}_{H} (x_k - x_k^R)$$

Assuming nominal measurements $z_k^R = h(x_k^R, k) + v$, we have

$$\delta z = z - z_k^R$$

$$= h(x_k^R, k) + H\delta x - h(x_k^R, k)$$

$$\boxed{\delta z = H\delta x}$$

where

$$H = \frac{\partial h(x,k)}{\partial x}$$

$$= \begin{bmatrix} \frac{\partial h_1(x,k)}{\partial x} & \frac{\partial h_1(x,k)}{\partial y} & \frac{\partial h_1(x,k)}{\partial \theta} & \frac{\partial h_1(x,k)}{\partial r_x^1} & \frac{\partial h_1(x,k)}{\partial r_y^1} & \frac{\partial h_1(x,k)}{\partial r_x^2} & \frac{\partial h_1(x,k)}{\partial r_y^2} \\ \vdots & & & & & & \\ \frac{\partial h_4(x,k)}{\partial x} & \cdots & & & & & \frac{\partial h_4(x,k)}{\partial r_y^2} \end{bmatrix}$$

where $h_1(x,k) = z_r^1$

$h_2(x,k) = z_\theta^1$

$h_3(x,k) = z_r^2$

$h_4(x,k) = z_\theta^2$

We have

$$\frac{\partial h_1(x,k)}{\partial r^1_x} = \frac{\partial z^1_r}{\partial r^1_x} = \frac{\Delta x_1}{\sqrt{\Delta x_1^2 + \Delta y_1^2}} \qquad \frac{\partial z^2_r}{\partial r^2_x} = \frac{\Delta x_2}{\sqrt{\Delta x_2^2 + \Delta y_2^2}}$$

Similarly we have

$$\frac{\partial h_1(x,k)}{\partial r^1_y} = \frac{\partial z^1_r}{\partial r^1_y} = \frac{\Delta y_1}{\sqrt{\Delta x_1^2 + \Delta y_1^2}} \qquad \frac{\partial z^2_r}{\partial r^2_y} = \frac{\Delta y_2}{\sqrt{\Delta x_2^2 + \Delta y_2^2}}$$

$$\frac{\partial h_2(x,k)}{\partial r^1_x} = \frac{\partial z^1_\theta}{\partial r^1_x} = \frac{-\Delta y_1}{\Delta x_1^2 + \Delta y_1^2} \qquad \frac{\partial z^2_\theta}{\partial r^2_x} = \frac{-\Delta y_2}{\Delta x_2^2 + \Delta y_2^2}$$

$$\frac{\partial h_2(x,k)}{\partial r^1_y} = \frac{\partial z^1_\theta}{\partial r^1_y} = \frac{\Delta x_1}{\Delta x_1^2 + \Delta y_1^2} \qquad \frac{\partial z^2_\theta}{\partial r^2_y} = \frac{\Delta x_2}{\Delta x_2^2 + \Delta y_2^2}$$

Therefore, we have

$$\boxed{\delta z = H \delta x}$$

where H in given by the above 4×7 matrix.

```
clc;
clear all;
close all;
```

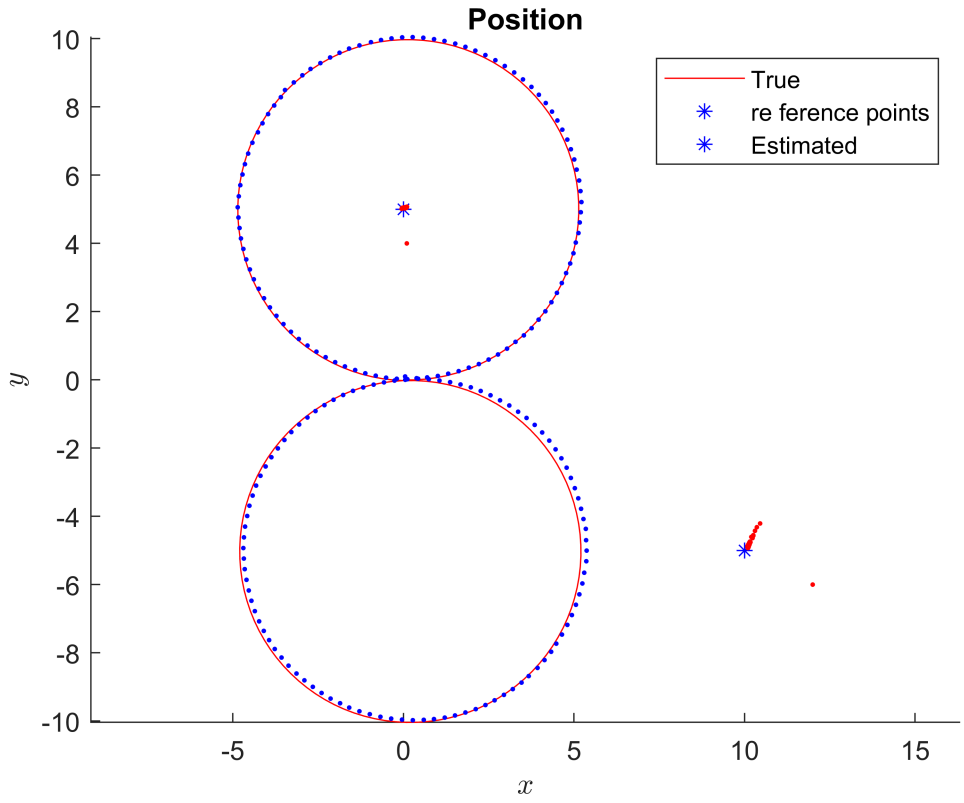
```
N=201;
t=linspace(0,20,N);
h=t(2)-t(1);
sigma_wV=1e-2;
sigma_wt=1e-2;
sigma_vr=1e-1;
sigma_vt=3*pi/180;

Nref=2;
ref=[0 10; 5 -5];
n=3+2*Nref;

Q=diag([sigma_wV^2 sigma_wt^2]);
R=diag([sigma_vr^2 sigma_vt^2 sigma_vr^2 sigma_vt^2]);

X0=[0.05 0.1 3*pi/180 ref(1,1)+0.1 ref(2,1)-1 ref(1,2)+2 ref(2,2)-1]';
P0=diag([0.2^2 0.2^2 (5*pi/180)^2 1^2 1^2 2^2 2^2]);

load prob3_wv;

X_true=zeros(n,N);
X_true(:,1)=[0 0 0 ref(1,1) ref(2,1) ref(1,2) ref(2,2)]';

V=pi;

% True Trajectory
for k=1:N-1
    if k <= (N-1)/2
        u(k)=pi/5;
    else
        u(k)=-pi/5;
    end
    theta_k=X_true(3,k);
    X_true(:,k+1)=X_true(:,k)...
        +[h*(V+w_V(k))*cos(theta_k);h*(V+w_V(k))*sin(theta_k);h*(u(k)+w_t(k));0;0;0;0];
end

% Measurement
for k=1:N
    x=X_true(1,k);
    y=X_true(2,k);
    theta=X_true(3,k);

    for j=1:Nref
        rx=X_true(2*j+2,k);
        ry=X_true(2*j+3,k);

        delx=rx-x;
        dely=ry-y;
```

```
        dis=sqrt(delx^2+dely^2);

        delP=[delx; dely];
        delP_body=[cos(theta) sin(theta);
            -sin(theta) cos(theta)]*delP;
        angle=atan2(delP_body(2),delP_body(1));

        z(2*j-1:2*j,k)=[dis; angle];
    end

    z(:,k)=z(:,k)+v(:,k);
end
```

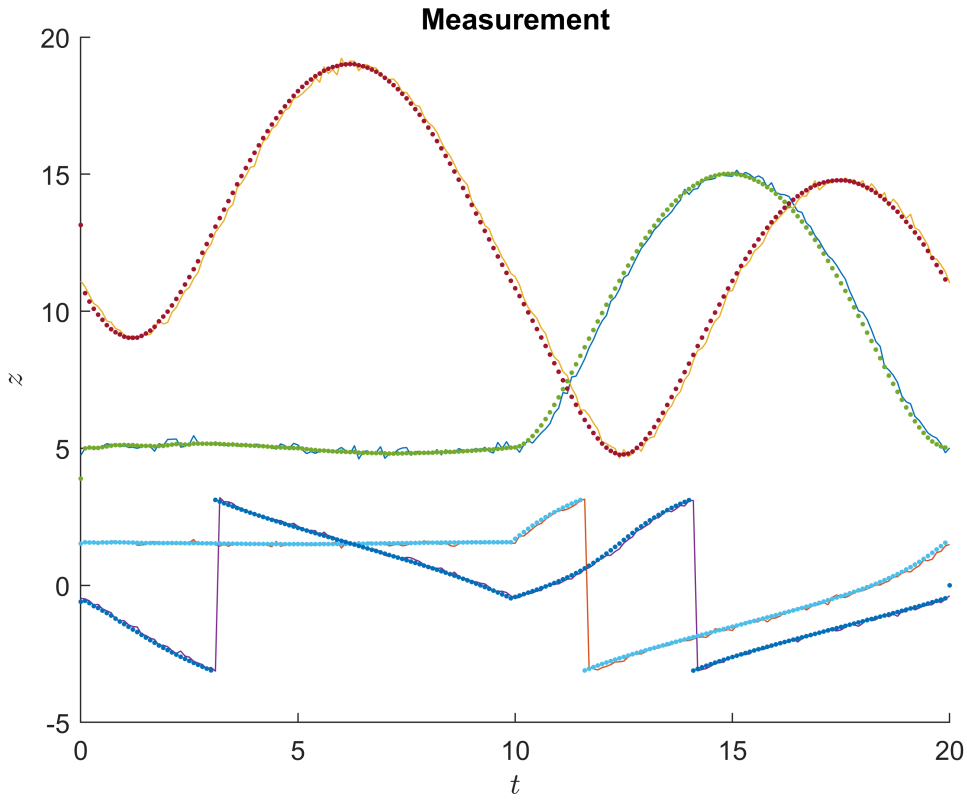
```
% EKF
X_bar=zeros(n,N);
P=zeros(n,n,N);
X_bar(:,1)=X0;
P(:,:,1)=P0;
Z_bar=zeros(2*Nref,N);
for l=1:N-1
    % prediction
    prev_x = X_bar(1,l);
    prev_y = X_bar(2,l);
    prev_theta = X_bar(3,l);
    prev_ref1x = X_bar(4,l);
    prev_ref1y = X_bar(5,l);
    prev_ref2x = X_bar(6,l);
    prev_ref2y = X_bar(7,l);

    curr_x = prev_x + h*V*cos(prev_theta);
    curr_y = prev_y + h*V*sin(prev_theta);
    curr_theta = prev_theta + h*u(l);
    curr_ref1x = prev_ref1x;
    curr_ref1y = prev_ref1y;
    curr_ref2x = prev_ref2x;
    curr_ref2y = prev_ref2y;
    temp_Ak=[1 0 -h*V*sin(prev_theta);
        0 1 h*V*cos(prev_theta);
        0 0 1];
    Ak=[temp_Ak zeros(3,2*Nref);
        zeros(2*Nref,3) eye(2*Nref)];
    Bk = [0; 0; h; 0; 0; 0; 0];
    Gk = [h*cos(prev_theta) 0; h*sin(prev_theta) 0; 0 h; 0 0; 0 0; 0 0; 0 0];
    X_bar_kp = [curr_x;curr_y;curr_theta;curr_ref1x;curr_ref1y;curr_ref2x;curr_ref2y];
    P_kp = Ak*P(:,:,l)*Ak' + Gk*Q*Gk';

    % Correction
    for n=1:Nref
        if n == 1
            refx = X_bar_kp(4);
            refy = X_bar_kp(5);
        else
```

```
            refx = X_bar_kp(6);
            refy = X_bar_kp(7);
        end
        delx = refx - curr_x;
        dely = refy - curr_y;
        dist = delx^2 + dely^2;
        % 2x1 matrix
        delP = [delx;dely];
        delP_body = [cos(curr_theta) sin(curr_theta);
            -sin(curr_theta) cos(curr_theta)]*delP;
        z_theta = atan2(delP_body(2), delP_body(1));
        % For each measurement, we have a 2x1 matrix
        Z_bar(2*n-1:2*n,1) = [sqrt(dist);z_theta];
        % 4x7 matrix
        dzdx = -delx/sqrt(dist);
        dzdy = -dely/sqrt(dist);
        dzdt = 0;
        dztdx = dely/dist;
        dztdy = -delx/dist;
        dztdt = -1;
        dzdrx = delx/sqrt(dist);
        dzdry = dely/sqrt(dist);
        dztdrx = -dely/dist;
        dztdry = delx/dist;
        H(:,:,n) = [dzdx dzdy dzdt dzdrx dzdry;
            dztdx dztdy dztdt dztdrx dztdry];
    end
    final_H = [H(:,:,1), zeros(2,2);
        H(1:2,1:3,2),zeros(2,2),H(1:2,4:5,2)];
    S = final_H*P_kp*final_H' + R;
    K = P_kp * final_H' / S;
    X_bar_kp = X_bar_kp + K*(z(:,l+1) - Z_bar(:,l));
    P_kp = (eye(7) - K*final_H)*P_kp;
    P(:,:,l+1) = P_kp;
    X_bar(:,l+1) = X_bar_kp;
end
```

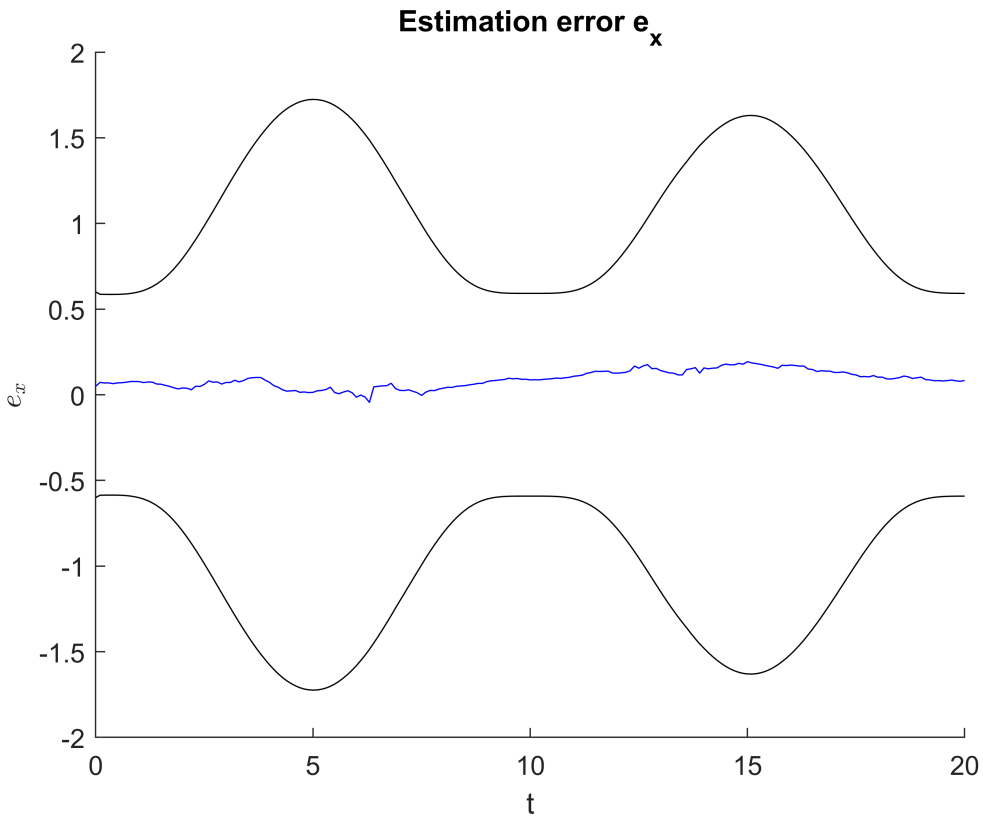
```
% Plot 1:
figure(1);
title('Position'); hold on;
plot(X_true(1,:),X_true(2,:),'r');
plot(X_true(4,:),X_true(5,:),'b*');
plot(X_true(6,:),X_true(7,:),'b*');
plot(X_bar(1,:),X_bar(2,:),'b.')
plot(X_bar(4,:),X_bar(5,:),'r.')
plot(X_bar(6,:),X_bar(7,:),'r.')
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');
axis equal;
legend('True', 're ference points', 'Estimated')
hold off;
```



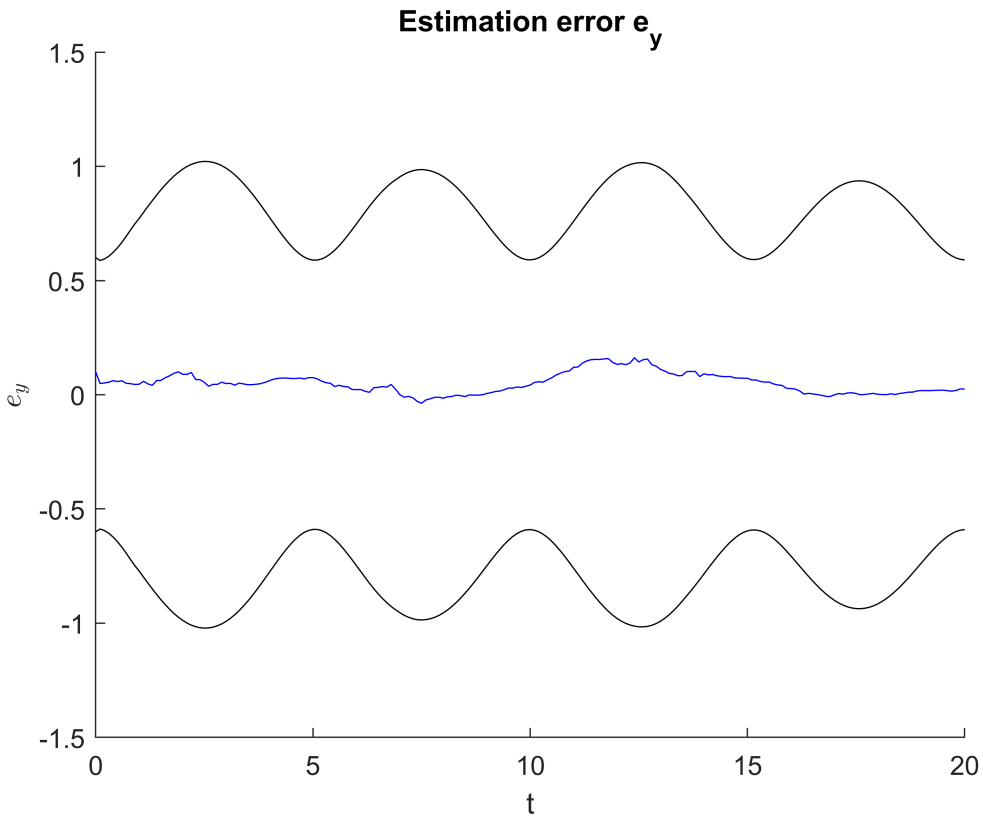

```
% Plot 2:
figure(2);
title("Measurement"); hold on;
plot(t,z);
plot(t,Z_bar, '.');
xlabel('$t$','interpreter','latex');
ylabel('$z$','interpreter','latex');
hold off
```




```
% Error Plotting
for m = 1:N
    sigma_x(m) = sqrt(P(1,1,m));
    sigma_y(m) = sqrt(P(2,2,m));
    sigma_theta(m) = sqrt(P(3,3,m));
end

% Plot 3:
diffX = X_bar(1,:) - X_true(1,:);
figure(3);
title('Estimation error e_x'); hold on;
plot(t,diffX, 'b',t,3*sigma_x,'k',t,-3*sigma_x,'k');
xlabel('t');
ylabel('$e_x$','interpreter','latex');
hold off;
```



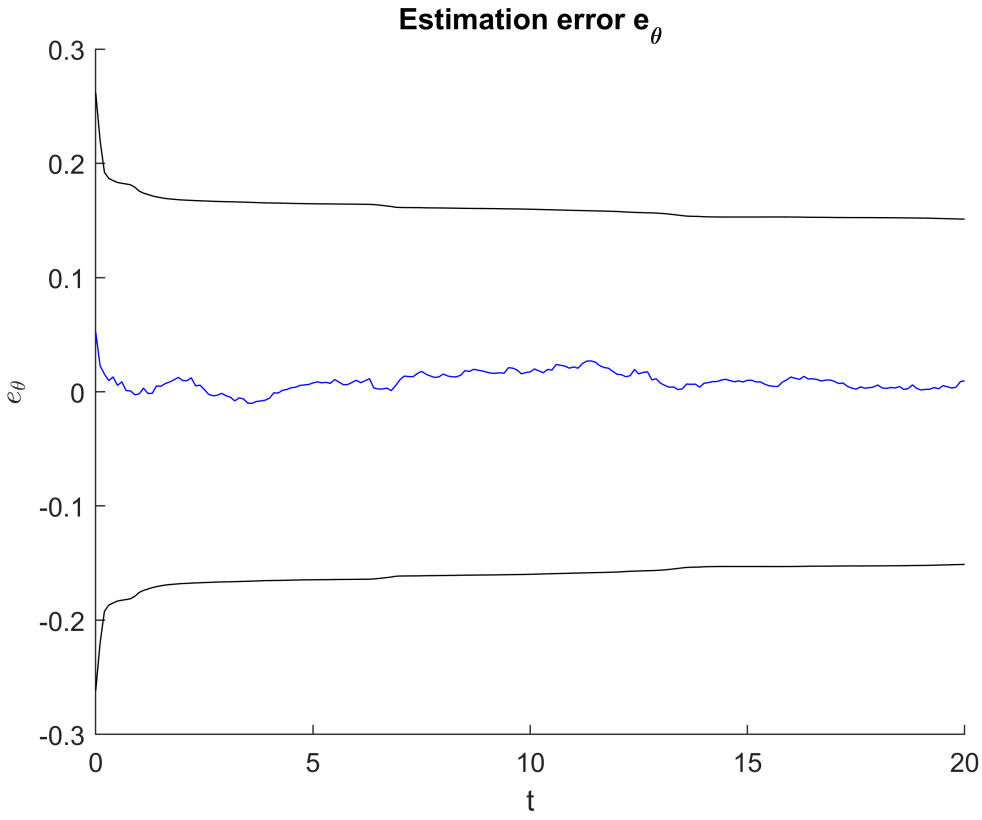

```
%Plot 4:
diffY = X_bar(2,:) - X_true(2,:);
figure(4);
title('Estimation error e_y'); hold on;
plot(t,diffY, 'b',t,3*sigma_y,'k',t,-3*sigma_y,'k');
xlabel('t');
ylabel('$e_y$','interpreter','latex');
hold off;
```



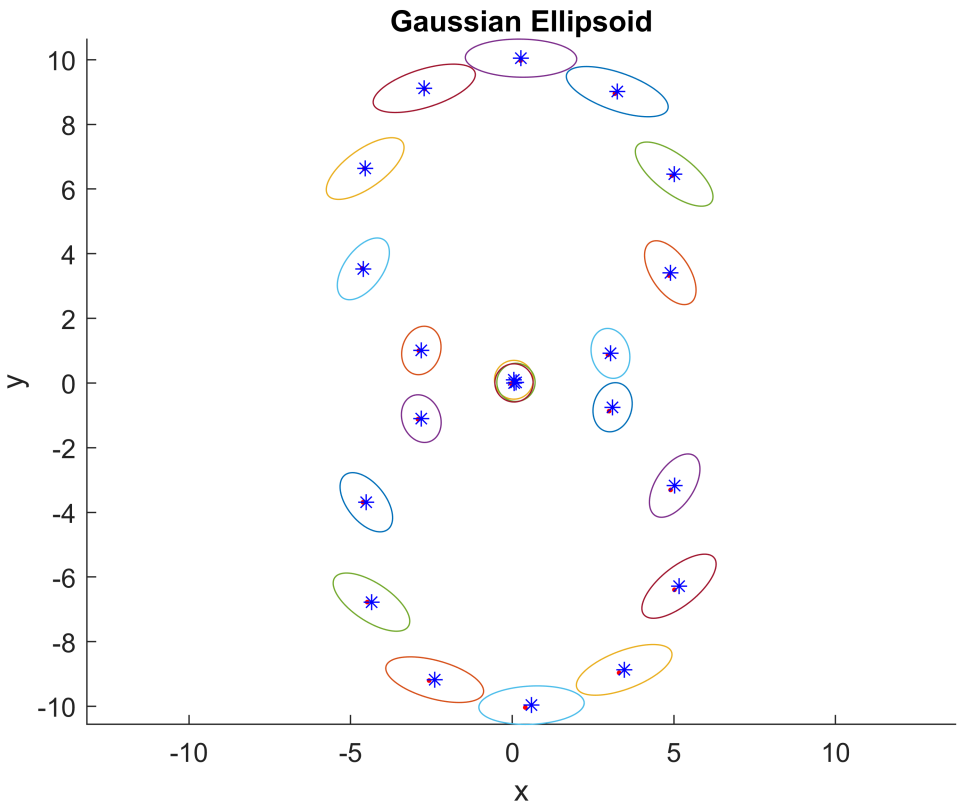

```
[[]]
```

```
ans =

     []
```

```
% Plot 5:
diffTheta = X_bar(3,:) - X_true(3,:);
figure(5);
title('Estimation error e_{\theta}'); hold on;
plot(t,diffTheta, 'b',t,3*sigma_theta,'k',t,-3*sigma_theta,'k');
xlabel('t');
ylabel('$e_\theta$','interpreter','latex');
hold off;
```




```
% Plot 6:
figure(6);
title('Gaussian Ellipsoid'); hold on;
for a=1:10:N
    plot(X_true(1,a), X_true(2,a),'r.');hold on;
    plot(X_bar(1,a), X_bar(2,a),'b*');
    plot_gaussian_ellipsoid(X_bar([1,2],a),P([1 2],[1,2],a),3);
end
axis equal;
xlabel('x');
ylabel('y');
hold off;
```


Gaussian Ellipsoid
10
8
6
4
2
0
-2
-4
-6
-8
-10
y
-10
-5
0
5
10
x

Problem 4 (Particle Filter for Localization) Consider the robot model discussed at Problem 1. Here we implement a particle filter as follows.

(a) Show that the state transition probability density function is given by

$$p(\mathbf{x}_{k+1}|\mathbf{x_k}, u_k) \sim \mathcal{N}(\begin{bmatrix} x_k + hV\cos\theta_k \\ y_k + hV\sin\theta_k \\ \theta_k + hu_k \end{bmatrix}, G_k Q_k G_k^T)$$

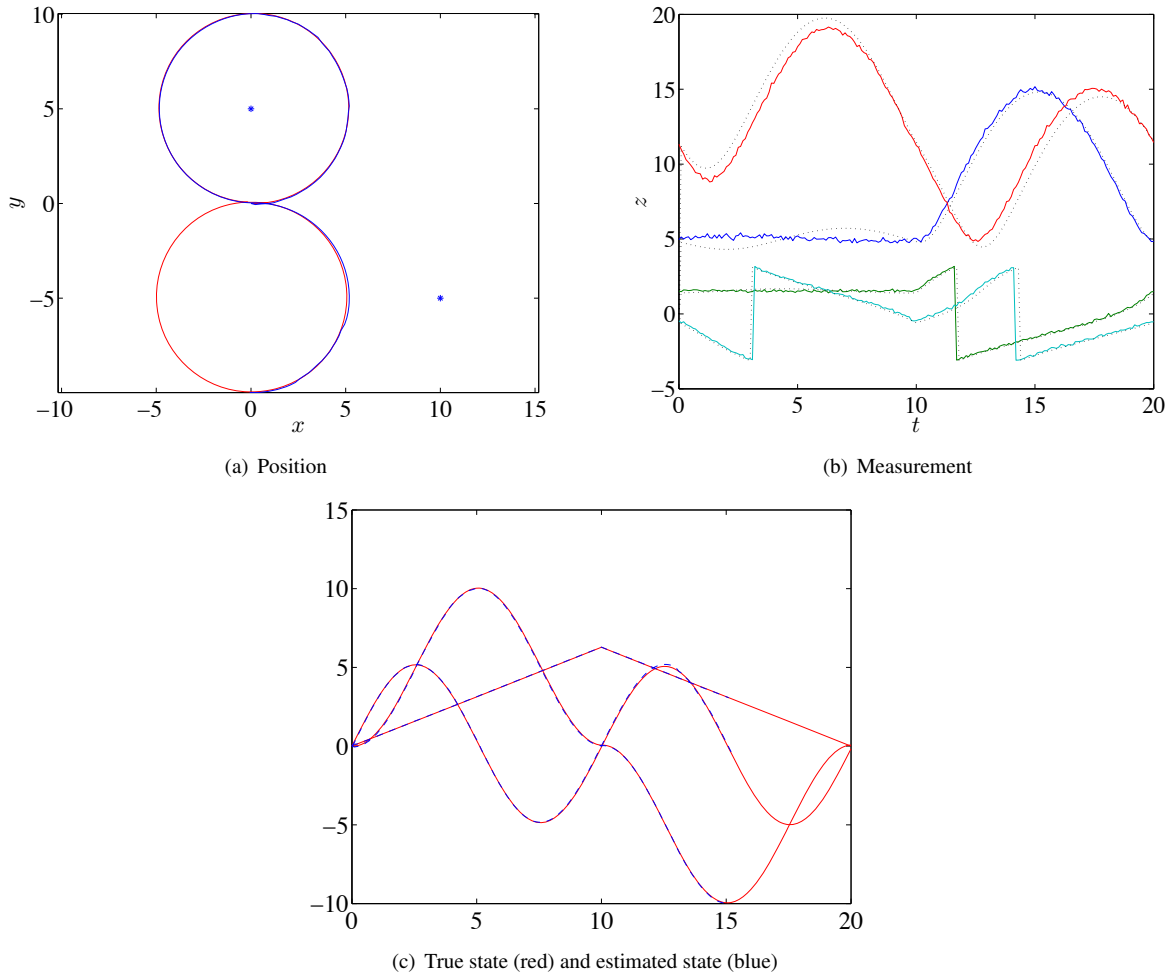
(b) Show that the sensor measurement probability density function for $\mathbf{z}^i = [z_r^i, z_\theta^i]^T \in \mathbb{R}^2$ is given by

$$p(\mathbf{z}^i|\mathbf{x}) = \mathcal{N}(\begin{bmatrix} \sqrt{\Delta x_i^2 + \Delta y_i^2} \\ \tan^{-1}\frac{\Delta y_i^{body}}{\Delta x_i^{body}} \end{bmatrix}, R_k)$$

(c) All of the parameters for the robot model, control inputs, measurements, reference points, and initial guess are specified at `prob4.m`. This also includes Matlab codes for the prediction step of a particle filter. Complete the missing parts for the correction step, and 3 plots given at the next page.
(Hint: you may use the Matlab `mvnpdf` function to evaluate a multivariate Gaussian pdf).

(d) The mean value, `X_bar` becomes NaN when $t \geq 15$ or so. Discuss the reason for this.

(e) Find, study, and summarize any resampling method for particle filters. Write a pseudo-code for the selected resampling algorithm.



(a) Position

(b) Measurement

(c) True state (red) and estimated state (blue)

Figure 4: Particle Filter for Localization

PROBLEM 4: Particle filter for Localization

Given equations of motion:

$$x_{k+1} = x_k + hV\cos\theta_k + hw_{V_k}\cos\theta_k$$

$$y_{k+1} = y_k + hV\sin\theta_k + hw_{V_k}\sin\theta_k$$

$$\theta_{k+1} = \theta_k + hu_k + hw_{\theta_k}$$

(a)

Rewriting the above equations, we have

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} x_k + hV\cos\theta_k \\ y_k + hV\sin\theta_k \\ \theta_k + hu_k \end{bmatrix}}_{f(x_k, u_k)} + \begin{bmatrix} h\cos\theta_k & 0 \\ h\sin\theta_k & 0 \\ 0 & h \end{bmatrix} \begin{bmatrix} wV_k \\ w\theta_k \end{bmatrix}$$

Therefore, we have

$$\boxed{X_{k+1} = f(x_k, u_k) + G_k w_k} \quad \text{— (1)}$$

- X_{k+1}: state at next timestep
- $f(x_k, u_k)$: function of curr. state & control input
- $G_k w_k$: process noise (w_k) is gaussian $\sim N(0, G_k Q_k G_k^T)$

State Transition probability is defined as

$p(x_{k+1} \mid x_k, u_k)$ where x_k is a random variable which is being used as a conditional variable. In this problem, x_k is a constant. u_k is also fixed

So, we have

$f(x_k, u_k)$ where x_k & u_k are fixed. So this function value is also constant considering the mean and covariance of the (1), we have

$$p(x_{k+1}|x_k,u_k) \sim N\left(\begin{bmatrix} x_k + hV\sin\theta_k \\ y_k + hV\cos\theta_k \\ \theta_k + hu_k \end{bmatrix}, G_k Q_k G_k^T\right)$$

mean of a gaussian distribution

covariance of ① [constant & $G_k w_k$]

$$p(x_{k+1}|x_k,u_k) = N\left(\begin{bmatrix} x_k + hV\sin\theta_k \\ y_k + hV\cos\theta_k \\ \theta_k + hu_k \end{bmatrix}, G_k Q_k G_k^T\right)$$

State transition probability density function

(b) Sensor measurement probability density function

We have

$$z^i = \begin{bmatrix} z_r^i \\ z_\theta^i \end{bmatrix} = \underbrace{\begin{bmatrix} \sqrt{\Delta x_i^2 + \Delta y_i^2} \\ \tan^{-1}\dfrac{\Delta y_i^{body}}{\Delta x_i^{body}} \end{bmatrix}}_{h^i(x)} + \begin{bmatrix} V_r \\ V_\theta \end{bmatrix}$$

↳ ∵ — measurement noise following Gaussian distribution

$$z^i = h^i(x) + V_k \qquad V \sim N(0, R_k)$$

We know that value of the state x in given → Hence it is not a stochastic variable. A constant added to a variable following gaussian distribution, then the sum is gaussian with mean = $h^i(x)$ and covariance = R_k

$$p(z^i|x) \sim N\left(\begin{bmatrix} \sqrt{\Delta x_i^2 + \Delta y_i^2} \\ \tan^{-1}\dfrac{\Delta y_i^{body}}{\Delta x_i^{body}} \end{bmatrix}, R_k\right)$$

Hence

Sensor measurement probability density function is

$$p(z^i|x) \sim \mathcal{N}\left(\begin{bmatrix} \sqrt{\Delta x_i^2 + \Delta y_i^2} \\ \tan^{-1} \dfrac{\Delta y_i^{body}}{\Delta x_i^{body}} \end{bmatrix}, R_k\right)$$

(C) mean value;

x_bar becomes NAN when $t \geq 15$

The reason for this is because $\bar{x}$ (mean) is calculated as $\Sigma w^i x^i$. After a number of iterations, the weights of most particle tend to go to negligible values whereas the weight of only few particles have really large weights. This reducing of the number of particles that are effective contributer to $\bar{x}$ (mean) value going as NaN A good way to avoid this is to use resampling methods.

(e) Resampling method for particle filters

① Select with replacement :

* simplest method
* select each particle with a probability equal to its weight.

Summary of steps:

① Cumulative sum of the particle weights are calculated.

② 'N' random numbers uniformly distributed in [0,1] are selected and sorted.

③ The number of sorted random numbers that appear in each interval of the cumulative sum represents the number of copies of that particular particle to be propagated to the next step

i.e.

* If a particle has a small weight, the equivalent cumulative sum interval is small → Fewer chances of a random number appearing in it
* If the weight of a particle is large, multiple number of random numbers are assigned to it. This process helps in making duplicates of that particle to be propagated to the next step.

Pseudo-code:

Input: weights of the particle.

Condition: $\sum_{i=1}^{N} w_i = 1$

Algorithm: Pseudocode:

```
Qsum = cumsum(W); { running totals Qsum = sum_{i=0}^{j} w_i }
rand_int = rand(N+1); { rand_int is an array of
                        N+1 random numbers }
T = sort(rand_int);
T(N+1) = 1 ; i = 1 ; j = 1
while (i ≤ N) do
    if T[i] < Q[j] then
        Index[i] = j;
        i = i+1;
    else
        j = j+1;
    end if
end while
Return (Index)
```

```
clc;
clear all;
close all;
```

```
N=201;
t=linspace(0,20,N);
h=t(2)-t(1);
sigma_wV=1e-2;
sigma_wt=1e-2;
sigma_vr=1e-1;
sigma_vt=3*pi/180;

Nref=2;
ref=[0 10; 5 -5];
n=3;

Q=diag([sigma_wV^2 sigma_wt^2]);
R=diag([sigma_vr^2 sigma_vt^2 sigma_vr^2 sigma_vt^2]);

X0=[0.05 0.1 3*pi/180]';
P0=diag([0.2^2 0.2^2 (5*pi/180)^2]);

load prob1_wv;

X_true=zeros(3,N);
X_true(:,1)=[0 0 0]';

V=pi;

% True Trajectory
for k=1:N-1
    if k <= (N-1)/2
        u(k)=pi/5;
    else
        u(k)=-pi/5;
    end
    theta_k=X_true(3,k);
    X_true(:,k+1)=X_true(:,k)...
        +[h*(V+w_V(k))*cos(theta_k);h*(V+w_V(k))*sin(theta_k);h*(u(k)+w_t(k))];
end



% Measurement
for k=1:N
    x=X_true(1,k);
    y=X_true(2,k);
    theta=X_true(3,k);

    for j=1:Nref
        rx=ref(1,j);
        ry=ref(2,j);
```

```
        delx=rx-x;
        dely=ry-y;

        dis=sqrt(delx^2+dely^2);

        delP=[delx; dely];
        delP_body=[cos(theta) sin(theta);
            -sin(theta) cos(theta)]*delP;
        angle=atan2(delP_body(2),delP_body(1));

        z(2*j-1:2*j,k)=[dis; angle];
    end

    z(:,k)=z(:,k)+v(:,k);
end
```

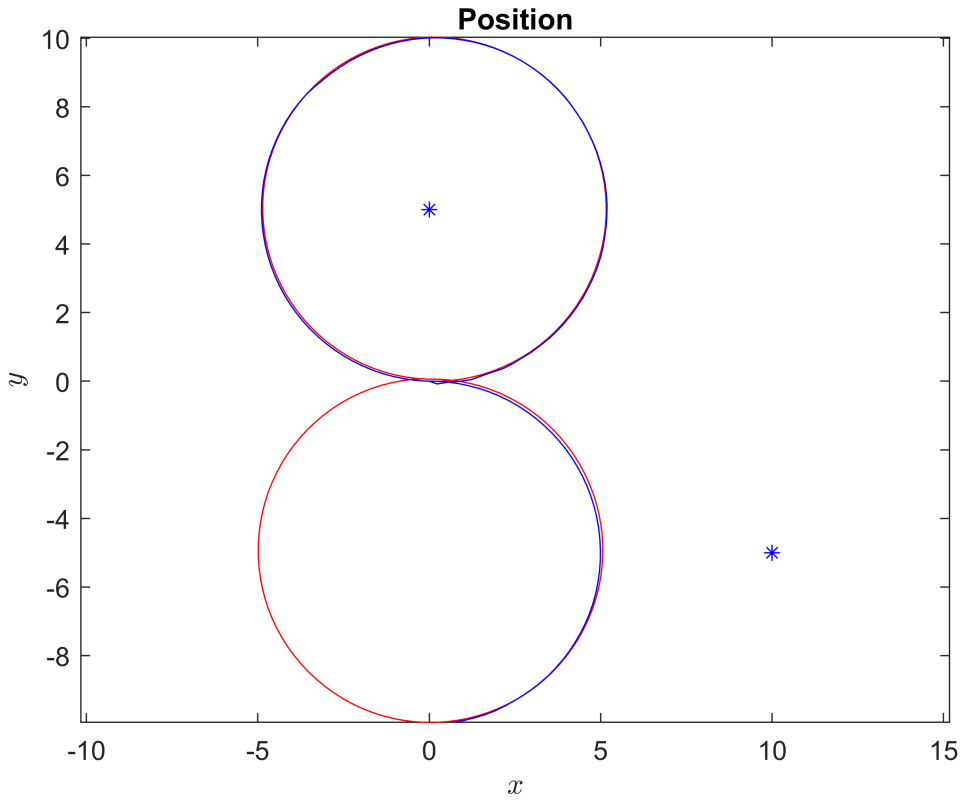
```
% Particle Filter

NN=1000;
XX_k=zeros(n,NN);
WW_k=zeros(NN,1);
XX_kp=zeros(n,NN);
WW_kp=zeros(NN,1);
X_bar=zeros(n,N);

for i=1:NN
    XX_k(:,i)=mvnrnd(X0,P0);
    WW_k(i)=1;
end
for i=1:NN
    X_bar(:,k)=X_bar(:,k)+WW_k(i)*XX_k(:,i);
end
X_bar(:,k)=X_bar(:,k)./sum(WW_k);


for k=1:N-1

    % prediction
    for i=1:NN
        theta_k=XX_k(3,i);
        XX_kp_bar=XX_k(:,i)+...
            +[h*V*cos(theta_k);h*V*sin(theta_k);h*u(k)];
        Gk=[h*cos(theta_k) 0;h*sin(theta_k) 0;0 h];
        PP_kp=Gk*Q*Gk';
        XX_kp(:,i)=mvnrnd(XX_kp_bar,PP_kp);
    end

    % correction
    for i=1:NN
        curr_x = XX_kp(1,i);
        curr_y = XX_kp(2,i);
        curr_theta = XX_kp(3,i);
        for n=1:Nref
```

```
            refx = ref(1,n);
            refy = ref(2,n);
            delx = refx - curr_x;
            dely = refy - curr_y;
            dist = sqrt(delx^2 + dely^2);
            delP = [delx;dely];
            delP_body = [cos(curr_theta) sin(curr_theta);
                -sin(curr_theta) cos(curr_theta)]*delP;
            z_theta = atan2(delP_body(2), delP_body(1));
            Z_bar(2*n-1:2*n,i) = [dist z_theta];
        end
        WW_kp(i)=WW_k(i)*mvnpdf(z(:,k+1), Z_bar(:,i),R);
    end

for i=1:NN
    X_bar(:,k+1)=X_bar(:,k+1)+WW_kp(i)*XX_kp(:,i);
end
X_bar(:,k+1)=X_bar(:,k+1)./sum(WW_kp);

XX_k=XX_kp;
WW_k=WW_kp;

if rem(k,10)==0
    disp(k/N);
end
end
```

```
   0.049751243781095

   0.099502487562189

   0.149253731343284

   0.199004975124378

   0.248756218905473

   0.298507462686567

   0.348258706467662

   0.398009950248756

   0.447761194029851

   0.497512437810945

   0.547263681592040

   0.597014925373134

   0.646766169154229

   0.696517412935323

   0.746268656716418

   0.796019900497512

   0.845771144278607
```

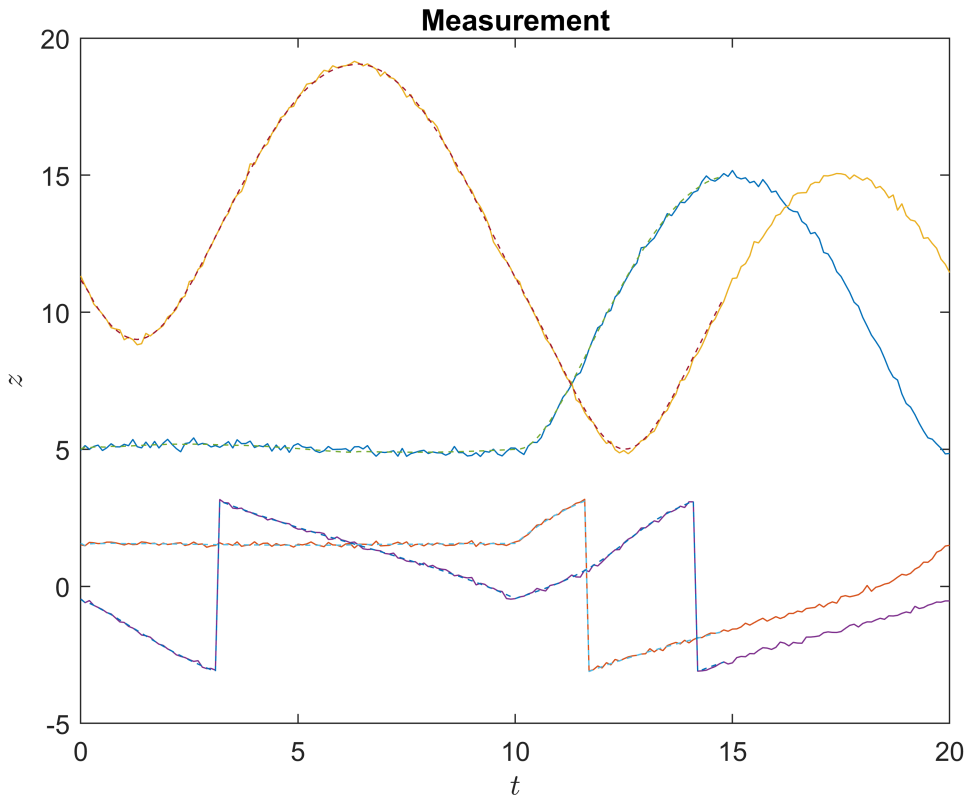
```
0.895522388059702

0.945273631840796

0.995024875621891
```

```
figure(1);
plot(X_true(1,:),X_true(2,:),'r');
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');
axis equal;
hold on;
title('Position')
plot(X_bar(1,:),X_bar(2,:),'b');
plot(ref(1,:),ref(2,:),'b*');
hold off
```

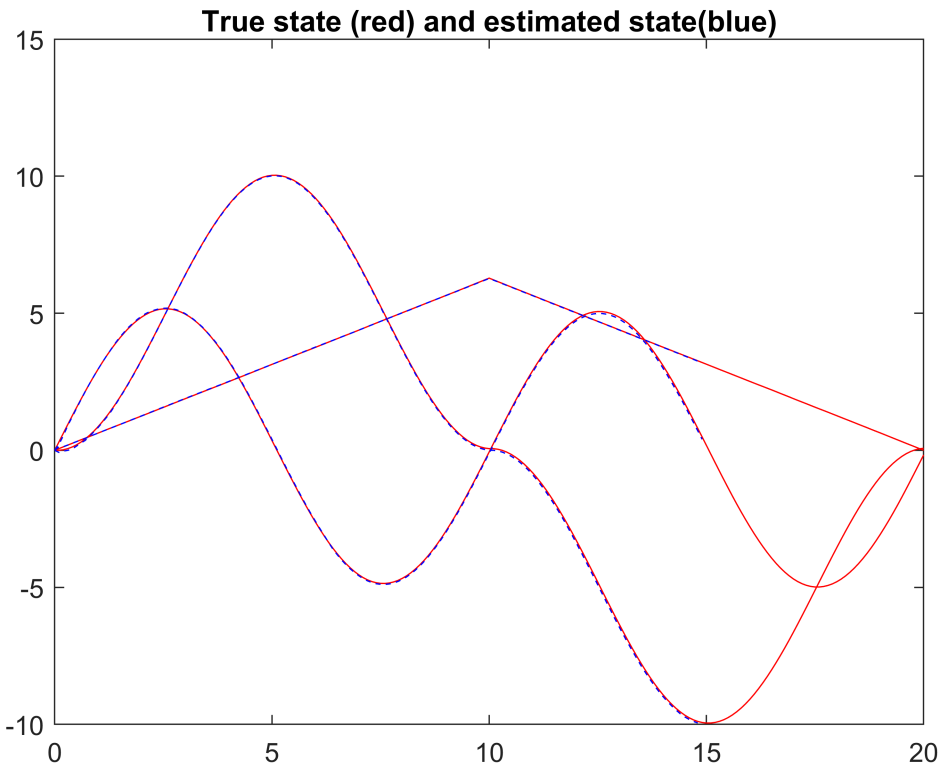



```
for i=1:N
    x = X_bar(1,i);
    y = X_bar(2,i);
    theta = X_bar(3,i);
    for n=1:Nref
        refx = ref(1,n);
        refy = ref(2,n);
        delx = refx - x;
```

```
        dely = refy - y;
        dist = sqrt(delx^2 + dely^2);
        delP = [delx;dely];
        delP_body = [cos(theta) sin(theta);
            -sin(theta) cos(theta)]*delP;
        z_theta = atan2(delP_body(2), delP_body(1));
        Z_est(2*n-1:2*n,i) = [dist; z_theta];
    end
end
figure(2);
plot(t,z); hold on
title('Measurement')
plot(t,Z_est,'--');
xlabel('$t$','interpreter','latex');
ylabel('$z$','interpreter','latex');
```




```
figure(3);
plot(t,X_true,'r',t,X_bar,'b--');
hold on;
title('True state (red) and estimated state(blue)')
```


True state (red) and estimated state(blue)
15
10
5
0
-5
-10
0
5
10
15
20