

ME 425 HW 4
Image Processing and Corner Detection On
Various Images In MATLAB

Arda Gencer
34124
Instructor: Mustafa Ünel

Fall 2025-2026
Due Date: 29.12 .2025



1 Introduction

In HW5 of ME425, the state estimation of a holonomic point robot navigating in a planar environment using a Kalman Filter (KF) was made. The robot's pose is described by the state vector $q = [x, y]$, while observations consist of the robot's distance and bearing angle relative to the origin, both subject to additive Gaussian noise. Simulation data is generated in Matlab/Simulink by applying control inputs along the x and y axes, with motion uncertainty modeled as Gaussian noise. The KF is implemented to provide continuous estimates of the robot's state. Key results include plots of predicted and updated states, along with the evolution of the a priori and a posteriori error covariance matrices, which are analyzed to assess filter performance.

The study is further extended to a non-holonomic point robot characterized by the state vector $q = [x, y, \theta]$. An Extended Kalman Filter (EKF) is developed to estimate the robot's pose under similar noise assumptions in both motion and measurement. Simulation outcomes and corresponding analyses demonstrate the EKF's effectiveness in tracking the robot's position and orientation over time.

2 Procedure

2.1 Question 1

The equations that were implemented in MATLAB to model the system and develop the Kalman Filter are as follows:

2.1.1 Modeling The State And Measurements

State (motion) model: $q_{k+1} = A q_k + B u_k + w_k, \quad w_k \sim \mathcal{N}(0, Q)$

Where $q = \begin{bmatrix} x \\ y \end{bmatrix}$, $u = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$, w_k is the process noise

Measurement model: $z_k = h(q_k) + v_k, \quad v_k \sim \mathcal{N}(0, R)$

with $h(q) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan2}(y, x) \end{bmatrix}$, $q = \begin{bmatrix} x \\ y \end{bmatrix}$

Jacobian (linearization of h): $H_k = \frac{\partial h}{\partial q} \Big|_{q=\hat{q}_{k|k-1}} = \begin{bmatrix} \frac{x}{\sqrt{x^2 + y^2}} & \frac{y}{\sqrt{x^2 + y^2}} \\ -\frac{y}{x^2 + y^2} & \frac{x}{x^2 + y^2} \end{bmatrix} \Big|_{q=\hat{q}_{k|k-1}}$

2.1.2 Kalman / Extended Kalman Filter Equations

Predict the state ahead: $\hat{q}_k^- = \hat{q}_{k-1} + \Delta t \ u_{k-1}$

Predict the error covariance ahead: $P_k^- = P_{k-1} + Q$

Compute the Kalman gain: $K_k = P_k^- H_k^\top (H_k P_k^- H_k^\top + R)^{-1}$

Update estimate with measurement z_k : $\hat{q}_k = \hat{q}_k^- + K_k (z_k - h(\hat{q}_k^-))$

Update the error covariance: $P_k = (I - K_k H_k) P_k^-$

2.2 Question 2

Consider a robot whose pose at discrete time k is

$$q_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}.$$

Control inputs are the forward velocity v_k and yaw rate ω_k ,

$$u_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix},$$

and sampling interval is Δt (denoted T in code). The discrete-time motion (process) model with additive Gaussian process noise $w_k \sim \mathcal{N}(0, Q)$ is

$$q_{k+1} = f(q_k, u_k) + w_k = \begin{bmatrix} x_k + \Delta t v_k \cos \theta_k \\ y_k + \Delta t v_k \sin \theta_k \\ \theta_k + \Delta t \omega_k \end{bmatrix} + w_k.$$

Measurements

At each time step we measure range and bearing from the origin:

$$z_k = \begin{bmatrix} d_k \\ \alpha_k \end{bmatrix} = h(q_k) + v_k, \quad v_k \sim \mathcal{N}(0, R),$$

with

$$h(q) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan2}(y, x) \end{bmatrix}.$$

Assume R is diagonal since the range and bearing noises are independent:

$$R = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix}.$$

Jacobians

Linearize about the predicted state $\hat{q}_{k|k-1} = [\hat{x}, \hat{y}, \hat{\theta}]^\top$.

Process-model Jacobian (w.r.t. state), often denoted F_k or A_k :

$$A_k \equiv \left. \frac{\partial f}{\partial q} \right|_{\hat{q}_{k|k-1}, u_{k-1}} = \begin{bmatrix} 1 & 0 & -\Delta t v_{k-1} \sin \hat{\theta} \\ 0 & 1 & \Delta t v_{k-1} \cos \hat{\theta} \\ 0 & 0 & 1 \end{bmatrix}.$$

Control-input Jacobian (w.r.t. u), denoted B_k :

$$B_k \equiv \left. \frac{\partial f}{\partial u} \right|_{\hat{q}_{k|k-1}, u_{k-1}} = \begin{bmatrix} \Delta t \cos \hat{\theta} & 0 \\ \Delta t \sin \hat{\theta} & 0 \\ 0 & \Delta t \end{bmatrix}.$$

Measurement-model Jacobian H_k (w.r.t. state) evaluated at $\hat{q}_{k|k-1}$: let $\rho = \sqrt{\hat{x}^2 + \hat{y}^2}$ and use a small ε to avoid division by zero if desired.

$$H_k \equiv \left. \frac{\partial h}{\partial q} \right|_{\hat{q}_{k|k-1}} = \begin{bmatrix} \frac{\hat{x}}{\rho} & \frac{\hat{y}}{\rho} & 0 \\ -\frac{\hat{y}}{\rho^2} & \frac{\hat{x}}{\rho^2} & 0 \end{bmatrix}, \quad \rho \equiv \sqrt{\hat{x}^2 + \hat{y}^2} + \varepsilon.$$

Extended Kalman Filter (EKF) equations

Denote the prior (predicted) state and covariance by \hat{q}_k^- and P_k^- , and the posterior (updated) state and covariance by \hat{q}_k and P_k .

1. Predict state (motion):

$$\hat{q}_k^- = f(\hat{q}_{k-1}, u_{k-1}) = \begin{bmatrix} \hat{x}_{k-1} + \Delta t v_{k-1} \cos \hat{\theta}_{k-1} \\ \hat{y}_{k-1} + \Delta t v_{k-1} \sin \hat{\theta}_{k-1} \\ \hat{\theta}_{k-1} + \Delta t \omega_{k-1} \end{bmatrix}.$$

2. Predict covariance:

$$P_k^- = A_k P_{k-1} A_k^\top + Q.$$

3. Compute innovation (measurement residual). For the angular residual use angle wrapping to keep the difference in $(-\pi, \pi]$:

$$\tilde{z}_k = z_k - h(\hat{q}_k^-) = \begin{bmatrix} d_k - \rho \\ \text{wrap}(\alpha_k - \text{atan2}(\hat{y}, \hat{x})) \end{bmatrix},$$

where $\rho = \sqrt{\hat{x}^2 + \hat{y}^2}$ and a convenient wrap function is $\text{wrap}(\phi) = \text{mod}(\phi + \pi, 2\pi) - \pi$.

4. Innovation covariance and Kalman gain:

$$S_k = H_k P_k^- H_k^\top + R, \quad K_k = P_k^- H_k^\top S_k^{-1}.$$

5. Update state estimate (apply the angle-wrapped innovation for the bearing component):

$$\hat{q}_k = \hat{q}_k^- + K_k \tilde{z}_k,$$

and ensure angle normalization:

$$\hat{\theta}_k \leftarrow \text{wrap}(\hat{\theta}_k).$$

6. Update covariance (Joseph form optional for numerical stability):

$$P_k = (I - K_k H_k) P_k^- (I - K_k H_k)^\top + K_k R K_k^\top$$

or (standard form)

$$P_k = (I - K_k H_k) P_k^-.$$

Implementation notes and practical tips

- Always keep the angle component wrapped to $[-\pi, \pi]$ immediately after prediction and after update; also wrap before using the angle in trigonometric functions.
- Use a small ε (e.g., 10^{-8}) when computing ρ to avoid division by zero in H_k .
- Tune Q and R to balance trust between the motion model and measurements. If the filter lags in θ :
 - increase Q_{33} (process noise variance for θ) or
 - decrease R_{22} (measurement noise variance for bearing).
- Enforce symmetry of P_k numerically: $P_k \leftarrow (P_k + P_k^\top)/2$ to reduce numerical drift.
- If EKF linearization fails (large nonlinearities), consider switching to an Unscented Kalman Filter (UKF).
- Plot the bearing innovation $\text{wrap}(\alpha_k - \text{atan2}(\hat{y}, \hat{x}))$ over time to debug slow convergence or bias.

3 Results

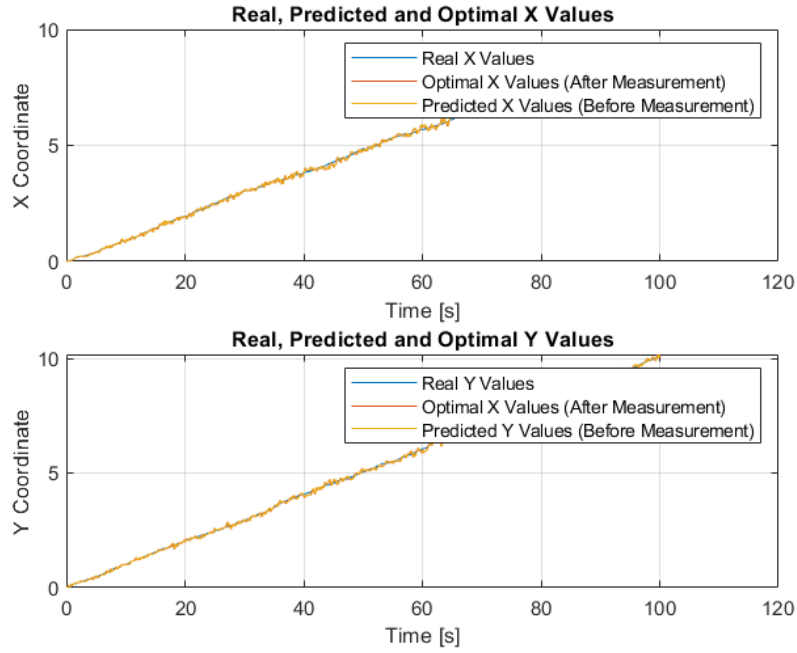


Figure 1: Question 1 Pose Values

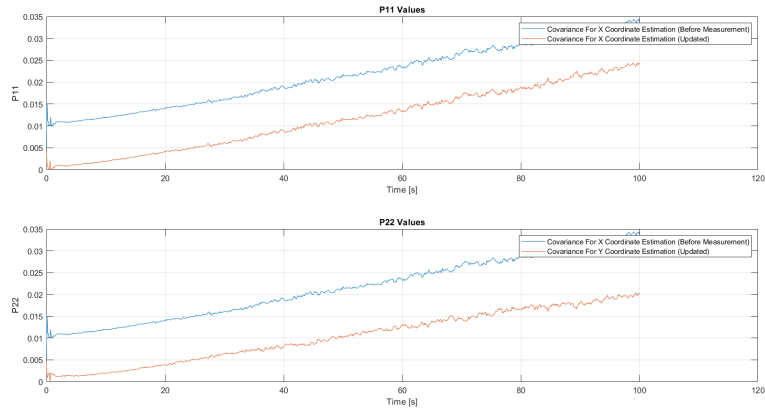


Figure 2: Question 1 Covariance Values

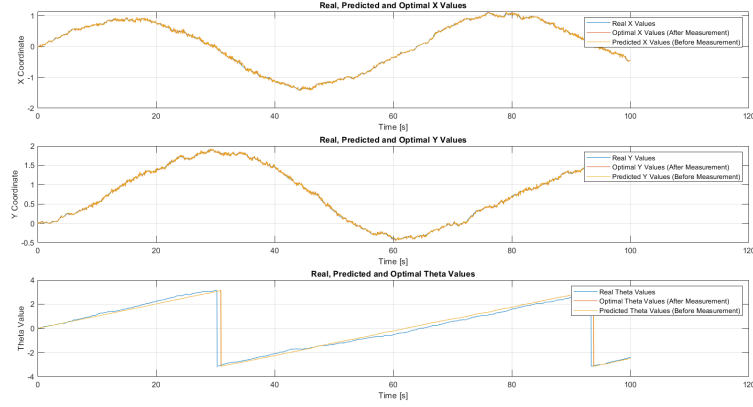


Figure 3: Question 2 Pose Values

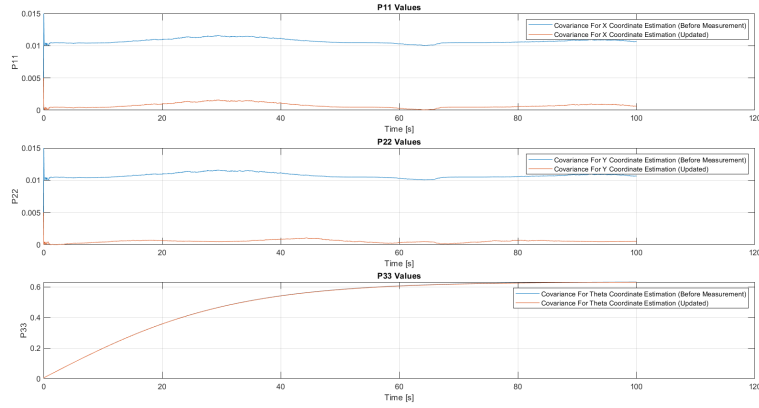


Figure 4: Question 2 Covariance Values

4 Discussion

In this work, state estimation for two types of planar robots were implemented and analyzed : a holonomic point robot with state vector $q = [x, y]^T$ using a standard Kalman Filter (KF), and a nonholonomic point robot with state vector $q = [x, y, \theta]^T$ using an Extended Kalman Filter (EKF). Both systems were observed through noisy measurements of range d and bearing α from the origin, with additive Gaussian noise assumed in both the motion and measurement models.

4.1 Holonomic Robot with Kalman Filter

The holonomic robot’s motion model is linear with independent control inputs along the x and y axes, allowing direct prediction of the state vector. The KF leverages this linearity to provide optimal state estimates under Gaussian noise assumptions.

The simulation results show that the predicted states x and y closely follow the true robot trajectory, with the a priori error covariance matrix elements decreasing over time as the filter gains confidence in its estimates. The a posteriori estimates further refine the state by incorporating measurements, resulting in reduced uncertainty reflected in the shrinking covariance values after each update.

The smooth convergence of the KF estimates and the monotonic decrease in covariance elements demonstrate the filter’s effectiveness in this linear, holonomic scenario. The direct control over both position coordinates simplifies the estimation problem, leading to fast and accurate tracking of the robot’s pose.

4.2 Nonholonomic Robot with Extended Kalman Filter

In contrast, the nonholonomic robot’s motion model is nonlinear due to the orientation θ and the constraint that motion occurs along the robot’s heading direction. The EKF accommodates this nonlinearity by linearizing the motion and measurement models around the current estimate.

Simulation results reveal that the EKF successfully tracks the robot’s pose (x, y, θ) over time, but with some notable differences compared to the holonomic case. The predicted states exhibit more variability, especially in the orientation θ , reflecting the increased complexity of the nonlinear model and the challenges in accurately linearizing the system dynamics.

The a priori error covariance matrix elements for the nonholonomic robot tend to be larger and decrease more slowly, indicating greater uncertainty in the predictions. After measurement updates, the a posteriori covariance reduces, but the filter requires careful tuning of process and measurement noise covariances to maintain stability and convergence.

The bearing measurement’s nonlinear dependence on the robot’s pose and the angle wrapping required for θ introduce additional challenges. These factors can cause slower convergence and occasional estimation lag, particularly in orientation, compared to the holonomic case.

4.3 Comparison and Insights

Overall, the KF for the holonomic robot demonstrates faster convergence, lower estimation error, and simpler implementation due to the linear system dynamics and direct control inputs. The EKF for the nonholonomic robot, while more complex, provides a practical solution for state estimation in constrained motion scenarios but requires more careful handling of nonlinearities and noise characteristics.

The results highlight the importance of model fidelity and noise tuning in nonlinear filtering. For the nonholonomic robot, improvements such as Unscented Kalman Filters (UKF) or particle filters could be explored to better handle nonlinearities and improve estimation accuracy.

5 Conclusion

The study confirms that while KF is well-suited for linear, holonomic systems, EKF is necessary for nonlinear, nonholonomic systems but comes with increased complexity and sensitivity. Proper noise modeling, angle wrapping, and Jacobian computation are critical for EKF performance. The simulation results and covariance analyses provide valuable insights into the behavior and limitations of these filters in robotic state estimation.

6 Appendix

Listing 1: The code for question 1

```

1  clc; clear; close all;
2
3  T = 0.1; %Sampling Time
4
5  t = 0; %Current Time
6  simTime = 100;
7
8  q_real = [1e-3;1e-3];
9  u = [0.1;0.1]; %Velocity Commands
10
11 A = eye(2);
12 q_old = q_real;
13 P_old = [0.005 0; 0 0.005];
14 real_covariance = [0.0001 0; 0 0.0001];
15 Q = [0.01 0; 0 0.01];
16 R = [0.0025 0; 0 0.0012];
17 %Data Recording Initialization
18 posRecord = [1e-3;1e-3];
19 estimateRecordBefore = [1e-3;1e-3];
20 estimateRecordAfter = [1e-3;1e-3];
21 times = [0];
22 P11_vals_before = [0.005];
23 P22_vals_before = [0.005];
24 P11_vals_after = [0.005];
25 P22_vals_after = [0.005];
26
27 while t < simTime
28     %Take Measurement
29     d = sqrt(q_real(1)^2 + q_real(2)^2);
30     alpha = atan2(q_real(2), q_real(1));
31     measurement_noise = transpose(mvnrnd(zeros(2,1), R));
32     measurement = [d; alpha] + measurement_noise;
33

```

```

34 %Extended Kalman Filter
35 q_before = q_old + T * u;
36 x = q_before(1);
37 y = q_before(2);
38 dist = x^2 + y^2;
39 if dist < 1e-4
40     dist = 1e-4;
41 end
42 H = [sqrt(dist)^-1*x sqrt(dist)^-1*y; -y/(dist) x/(dist)]; %
    Jacobian
43 P_before = A*P_old*transpose(A) + Q;
44 kalman_Gain = P_before*transpose(H) * inv(H* P_before*transpose
    (H)+R);
45 innovation = measurement- [sqrt(x^2+y^2); atan2(y,x)];
46 %Angle Wrapping
47 innovation(2) = mod(innovation(2) + pi, 2*pi) - pi;
48
49 q_after = q_before + kalman_Gain * (innovation);
50 P_after = (eye(size(kalman_Gain*H)) - kalman_Gain * H)*P_before
    ;
51
52
53 %Update Real Location
54 noise_real = transpose(mvnrnd(zeros(2,1), real_covariance));
55 q_real = q_real + [u(1) * T; u(2) * T] + noise_real;
56 %Update Time And Other Variables
57 t = t + T;
58 q_old = q_after;
59 P_old = P_after;
60
61
62 %Record Values
63 posRecord = [posRecord q_real];
64 estimateRecordBefore = [estimateRecordBefore q_before];
65 estimateRecordAfter = [estimateRecordAfter q_after];
66 times = [times t];
67 P11_vals_before = [P11_vals_before P_before(1,1)];
68 P22_vals_before = [P22_vals_before P_before(2,2)];
69 P11_vals_after = [P11_vals_after P_after(1,1)];
70 P22_vals_after = [P22_vals_after P_after(2,2)];
71 end
72
73 %% Plotting
74 figure;
75 subplot(2,1,1);
76 plot(times, posRecord(1,:), "DisplayName", "Real X Values");
77 title("Real, Predicted and Optimal X Values");
78 hold on;
79 plot(times, estimateRecordAfter(1,:), "DisplayName", "Optimal X
    Values (After Measurement)");
80 plot(times, estimateRecordBefore(1,:), "DisplayName", "Predicted X
    Values (Before Measurement)");
81 grid on;
82 legend show;
83 xlabel("Time [s]");
84 ylabel("X Coordinate");
85

```

```

86 subplot(2,1,2);
87 plot(times, posRecord(2,:), "DisplayName", "Real Y Values");
88 title("Real, Predicted and Optimal Y Values");
89 hold on;
90 xlabel("Time [s]");
91 ylabel("Y Coordinate");
92 plot(times, estimateRecordAfter(2,:), "DisplayName", "Optimal X
    Values (After Measurement)");
93 plot(times, estimateRecordBefore(2,:), "DisplayName", "Predicted Y
    Values (Before Measurement)");
94 grid on;
95 legend show;
96
97 figure;
98 subplot(2,1,1);
99 plot(times, P11_vals_before, "DisplayName", "Covariance For X
    Coordinate Estimation (Before Measurement)");
100 hold on;
101 plot(times, P11_vals_after, "DisplayName", "Covariance For X
    Coordinate Estimation (Updated)");
102 grid on;
103 title("P11 Values");
104 legend show;
105 xlabel("Time [s]");
106 ylabel("P11");
107
108 subplot(2,1,2);
109 plot(times, P11_vals_before, "DisplayName", "Covariance For X
    Coordinate Estimation (Before Measurement)");
110 hold on;
111 plot(times, P22_vals_after, "DisplayName", "Covariance For Y
    Coordinate Estimation (Updated)");
112 grid on;
113 title("P22 Values");
114 legend show;
115 xlabel("Time [s]");
116 ylabel("P22");

```

Listing 2: The code for question 2

```

1  clc; clear; close all;
2
3  T = 0.05; %Sampling Time
4
5  t = 0; %Current Time
6  simTime = 100;
7
8  q_real = [1e-3; 1e-3; 1e-3];
9  u = [0.1; 0.1]; %Velocity Commands
10
11 q_old = q_real;
12 P_old = [0.005 0 0; 0 0.005 0; 0 0 0.005];
13 real_covariance = [0.0001 0 0; 0 0.0001 0; 0 0 0.0001];
14 Q = [0.01 0 0; 0 0.01 0; 0 0 0.001];
15 R = [0.0005 0; 0 0.0005];
16 %Data Recording Initialization
17 posRecord = [1e-3; 1e-3; 1e-3];

```

```

18 estimateRecordBefore = [1e-3;1e-3; 1e-3];
19 estimateRecordAfter = [1e-3;1e-3; 1e-3];
20 times = [0];
21 P11_vals_before = [0.005];
22 P22_vals_before = [0.005];
23 P11_vals_after = [0.005];
24 P22_vals_after = [0.005];
25 P33_vals_before = [0.005];
26 P33_vals_after = [0.005];
27
28 while t < simTime
29     %Take Measurement
30     d = sqrt(q_real(1)^2 + q_real(2)^2);
31     alpha = atan2(q_real(2), q_real(1));
32     measurement_noise = transpose(mvnrnd(zeros(2,1), R));
33     measurement = [d; alpha] + measurement_noise;
34
35     %Extended Kalman Filter
36     B = [cos(q_old(3)) 0; sin(q_old(3)) 0; 0 1];
37     q_before = q_old + T * B * u;
38
39     x = q_before(1);
40     y = q_before(2);
41     theta = q_before(3);
42     theta = mod(theta + pi, 2*pi) - pi; % wrap to [-pi, pi]
43     innovation = measurement - [sqrt(x^2+y^2); atan2(y,x)];
44     %Angle Wrapping
45     if innovation(2) < -pi
46         innovation(2) = innovation(2) + 2*pi;
47     elseif innovation(2) > pi
48         innovation(2) = innovation(2) - 2*pi;
49     end
50     A = [1 0 -u(1)*sin(theta)*T; 0 1 u(1)*cos(theta)*T; 0 0 1];
51     H = [sqrt(x^2+y^2)^-1*x sqrt(x^2+y^2)^-1*y 0; -y/(x^2+y^2)
52          x/(x^2+y^2) 0]; %Jacobian
53     P_before = A*P_old*transpose(A) + Q;
54     kalman_Gain = P_before*transpose(H) * inv(H* P_before*
55         transpose(H)+R);
56     q_after = q_before + kalman_Gain * (innovation);
57     q_after(3) = mod(q_after(3) + pi, 2*pi) - pi; % wrap to [-pi, pi]
58     P_after = (eye(size(kalman_Gain*H)) - kalman_Gain * H)*
59         P_before;
60
61     %Update Real Location
62     noise_real = transpose(mvnrnd(zeros(3,1), real_covariance));
63     q_real = q_real + T * B * u + noise_real;
64     q_real(3) = mod(q_real(3) + pi, 2*pi) - pi; % wrap to [-pi, pi]
65     %Update Time And Other Variables
66     t = t + T;
67     q_old = q_after;
68     q_old(3) = mod(q_old(3) + pi, 2*pi) - pi; % wrap to [-pi, pi]
69     P_old = P_after;

```

```

68
69
70     %Record Values
71     posRecord = [posRecord q_real];
72     estimateRecordBefore = [estimateRecordBefore q_before];
73     estimateRecordAfter = [estimateRecordAfter q_after];
74     times = [times t];
75     P11_vals_before = [P11_vals_before P_before(1,1)];
76     P22_vals_before = [P22_vals_before P_before(2,2)];
77     P11_vals_after = [P11_vals_after P_after(1,1)];
78     P22_vals_after = [P22_vals_after P_after(2,2)];
79     P33_vals_before = [P33_vals_before P_before(3,3)];
80     P33_vals_after = [P33_vals_after P_after(3,3)];
81 end
82
83 %% Plotting
84 figure;
85 subplot(3,1,1);
86 plot(times, posRecord(1,:), "DisplayName", "Real X Values");
87 title("Real, Predicted and Optimal X Values");
88 hold on;
89 plot(times, estimateRecordAfter(1,:), "DisplayName", "Optimal X
    Values (After Measurement)");
90 plot(times, estimateRecordBefore(1,:), "DisplayName", "Predicted
    X Values (Before Measurement)");
91 grid on;
92 legend show;
93 xlabel("Time [s]");
94 ylabel("X Coordinate");
95
96 subplot(3,1,2);
97 plot(times, posRecord(2,:), "DisplayName", "Real Y Values");
98 title("Real, Predicted and Optimal Y Values");
99 hold on;
100 xlabel("Time [s]");
101 ylabel("Y Coordinate");
102 plot(times, estimateRecordAfter(2,:), "DisplayName", "Optimal Y
    Values (After Measurement)");
103 plot(times, estimateRecordBefore(2,:), "DisplayName", "Predicted
    Y Values (Before Measurement)");
104 grid on;
105 legend show;
106
107 subplot(3,1,3);
108 plot(times, posRecord(3,:), "DisplayName", "Real Theta Values");
109 title("Real, Predicted and Optimal Theta Values");
110 hold on;
111 xlabel("Time [s]");
112 ylabel("Theta Value");
113 plot(times, estimateRecordAfter(3,:), "DisplayName", "Optimal
    Theta Values (After Measurement)");
114 plot(times, estimateRecordBefore(3,:), "DisplayName", "Predicted
    Theta Values (Before Measurement)");
115 grid on;
116 legend show;
117
118

```

```

119
120     figure;
121     subplot(3,1,1);
122     plot(times,P11_vals_before,"DisplayName", "Covariance For X
        Coordinate Estimation (Before Measurement)");
123     hold on;
124     plot(times, P11_vals_after,"DisplayName","Covariance For X
        Coordinate Estimation (Updated)");
125     grid on;
126     title("P11 Values")
127     legend show;
128     xlabel("Time [s]");
129     ylabel("P11");
130
131     subplot(3,1,2);
132     plot(times,P11_vals_before,"DisplayName", "Covariance For Y
        Coordinate Estimation (Before Measurement)");
133     hold on;
134     plot(times,P22_vals_after,"DisplayName","Covariance For Y
        Coordinate Estimation (Updated)");
135     grid on;
136     title("P22 Values");
137     legend show;
138     xlabel("Time [s]");
139     ylabel("P22");
140
141     subplot(3,1,3);
142     plot(times,P33_vals_before,"DisplayName", "Covariance For Theta
        Coordinate Estimation (Before Measurement)");
143     hold on;
144     plot(times,P33_vals_after,"DisplayName","Covariance For Theta
        Coordinate Estimation (Updated)");
145     grid on;
146     title("P33 Values");
147     legend show;
148     xlabel("Time [s]");
149     ylabel("P33");

```