# 1 Part 1

Model (true state position $x(k)$) is given:

$$x(k+1) = x(k) + w(k) \tag{1}$$
$$y(k) = x(k) + v(k) \tag{2}$$

The model's stochastic elements, $w(k)$ and $v(k)$, are characterized as zero-mean, independent Gaussian random variables, with their respective standard deviations being $\sigma_w = 0.1$ and $\sigma_v = 1$. The true initial condition of the system is specified as $x(0) = 5$.
For the initialization of the Kalman filter, the initial estimate of the state is set to $\hat{x} = 0$. A significant level of uncertainty is attributed to this initial guess by assigning a large value to the initial state covariance, $P(0|-1) = 100000$.
The Kalman Gain, $K_f(k)$, and the state extrapolation gain, $K(k)$, are determined by applying Equations (3) and (4), respectively. Following this, the updated estimate uncertainty, $P(k|k)$, and the extrapolated uncertainty, $P(k+1|k)$, are calculated using Equations (5) and (6), completing the uncertainty propagation and update steps.

$$K_f(k) = P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1} \tag{3}$$
$$K(k) = (AP(k|k-1)C^T)(CP(k|k-1)C^T + R_2)^{-1} \tag{4}$$
$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1) \tag{5}$$
$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1 \tag{6}$$

Then use equation (7) and (8) to update estimate with measurement and extrapolate the state:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_f(k)(y(k) - C\hat{x}(k|k-1)) \tag{7}$$
$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{8}$$

The given discrete state-space model and the Kalman filter equations are implemented to perform the recursive calculations. The complete source code for this implementation is provided in **A.1**.
To visualize the filter's performance and convergence, the following results are plotted in the accompanying figures: Figure 1 presents a comparison of the True State $x(k)$ and the Filtered Estimate $\hat{x}(k|k)$ alongside the noisy measurements. Meanwhile, Figure 2,3 illustrates the evolution of key performance metrics, including the **Kalman Gain** $K_f(k)$ and the **Error Covariance** $P(k|k)$, both demonstrating the filter's convergence from the high initial uncertainty to a steady-state solution.
To quantitatively assess the performance of the Kalman filter, we calculate the average difference (Bias) and the Mean Squared Error (MSE) between the filtered estimate $\hat{x}(k|k)$ and the True State $x(k)$ across the entire time sequence.

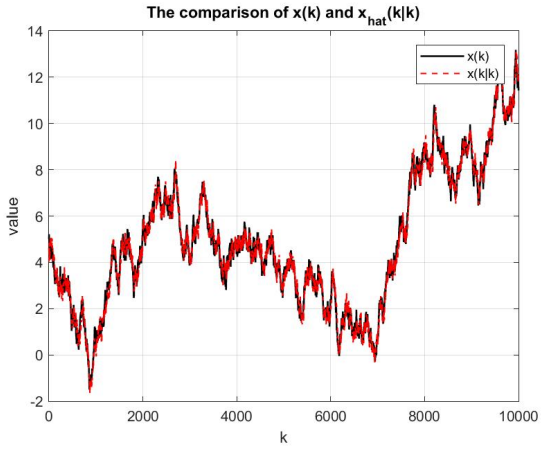


**Figure 1:** True state $x(k)$ (solid line) versus its filtered estimate $\hat{x}(k|k)$ (dotted line).

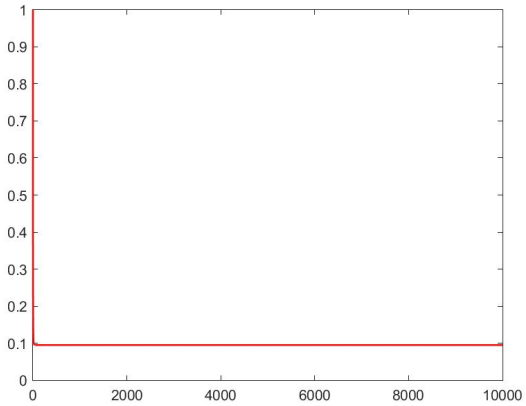


**Figure 2:** Updated error covariance $P(k|k)$ over time.

$$\text{Bias} = \frac{1}{N+1}\sum_{k=0}^{N}(x(k) - \hat{x}(k|k)) = 0.0034 \tag{9}$$

$$\text{Variance} = \frac{1}{N+1}\sum_{k=0}^{N}(x(k) - \hat{x}(k|k))^2 = 0.0923 \tag{10}$$

## 1.1 Analysis

As the fig shown, the convergence of the updated error covariance, $P(k|k)$, to a steady-state value of $P_{\text{steady}} \approx 0.1$ signifies that the Kalman filter has reached its **optimal, stable estimation limit**. Since the system is continuously corrupted by process and measurement noise ($\sigma_w$ and $\sigma_v$), the estimation error variance cannot be reduced to zero. The

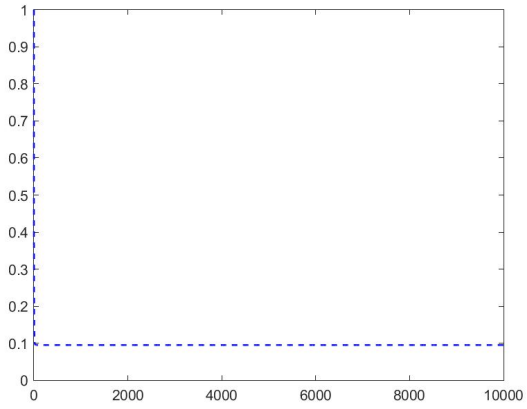


**Figure 3:** Kalman Gain $K_f(k)$ over time.

steady-state value $P_{\text{steady}}$ represents the minimum achievable estimation uncertainty under the given noise conditions. Notably, $P_{\text{steady}} = 0.1$ is one order of magnitude smaller than the measurement noise covariance $R_2 = 1$, confirming the filter's effectiveness in suppressing noise by leveraging the system dynamics.

The Kalman Gain, $K_f(k)$, converges to a steady-state value of $K_{\text{steady}} \approx 0.1$, a similar value. This gain dictates the weight given to the new measurement in the state update. A small convergence value implies that, in the steady-state, **only about** $10\%$ **of the innovation (the difference between the measurement and the prediction) is used to correct the state estimate**. This is a direct consequence of the chosen noise parameters: the process noise $\sigma_w = 0.1$ is much smaller than the measurement noise $\sigma_v = 1$. The filter thus correctly maintains a high degree of confidence in its own model-based prediction, while heavily rejecting the highly uncertain measurement data.

The calculated values for the Bias and MSE (Bias $= 0.0034$, MSE $= 0.0923$) demonstrate the filter's efficacy. By processing the raw, noisy measurements, the Kalman filter successfully minimizes the overall error signal and significantly reduces the variance of the state estimates. This performance is primarily attributed to the filter's ability to effectively leverage its reliable process model (characterized by a small process noise covariance, $R_1 = 0.01$) to suppress the substantial measurement noise (indicated by $R_2 = 1$).

# 2 Part 2

The system state is now a two-dimensional vector $\mathbf{x}(k) = [x_1(k), x_2(k)]^T$, where $x_1(k)$ is the target's position and $x_2(k)$ is its velocity.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} w(k) \tag{11}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v(k) \tag{12}$$

Where $w(k)$ and $v(k)$ are zero-mean independent Gaussian random variables with standard deviations $\sigma_w = 0.1$ and $\sigma_v = 1$ respectively. The sampling period is $T = 1$ and the initial conditions are $x_1(0) = 0$ and $x_2(0) = 30$.
The process noise covariance matrix $\mathbf{R}_1$ is calculated as:

$$\mathbf{R}_1 = \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} (0.01) \begin{bmatrix} 1/2 & 1 \end{bmatrix} = \begin{bmatrix} 0.0025 & 0.005 \\ 0.005 & 0.01 \end{bmatrix}$$

Similar to part 1, Kalmen Filter use equation (3), (4), (5), and (6) to calculate Kalman gain and estimate uncertainty by these equations; Then use equation (7) and (8) to estimated states using Kalman Filter.

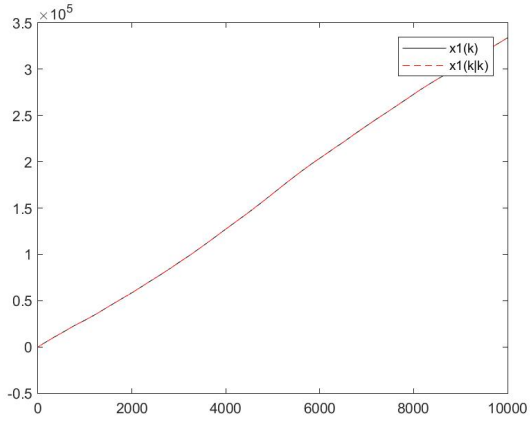


**Figure 4:** True state component $x_1(k)$ and its filtered estimate $\hat{x}_1(k|k)$.

The code for this implementation is provided in **A.2**. The required variables are plotted in Figures 4 to 9.
The overall biases and variances (MSE) are calculated for both position ($x_1$) and velocity ($x_2$) estimates:

$$\frac{1}{N+1} \sum_{k=0}^{N} (x_1(k) - \hat{x}_1(k|k)) = 0.0028 \tag{13}$$

$$\frac{1}{N+1} \sum_{k=0}^{N} (x_2(k) - \hat{x}_2(k|k)) = 0.0031 \tag{14}$$

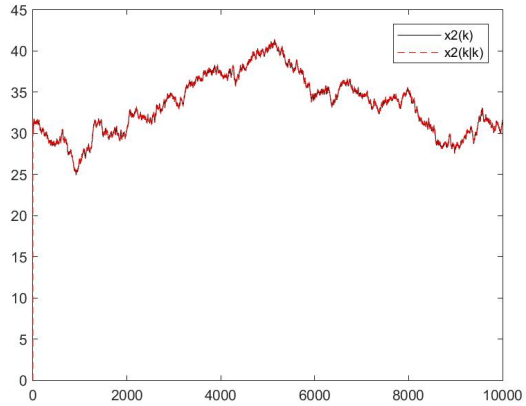


**Figure 5:** True state component $x_2(k)$ and its filtered estimate $\hat{x}_2(k|k)$.

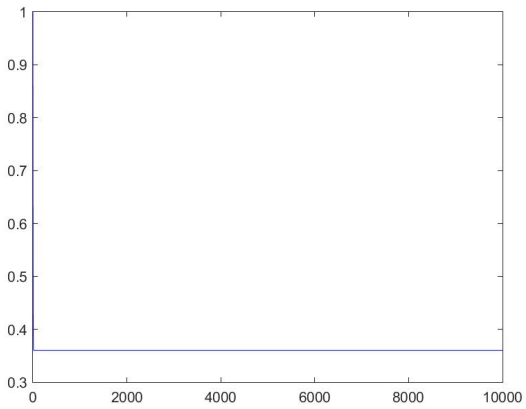


**Figure 6:** Updated error covariance for the position state, $P_{11}(k|k)$.

$$\frac{1}{N+1}\sum_{k=0}^{N}(x_1(k)-\hat{x}_1(k|k))^2 = 0.3618 \tag{15}$$

$$\frac{1}{N+1}\sum_{k=0}^{N}(x_2(k)-\hat{x}_2(k|k))^2 = 0.1315 \tag{16}$$

## 2.1 Analysis

The simulation results confirm the effectiveness of the two-state Kalman filter in tracking a moving target's position and velocity simultaneously, despite only measuring position.

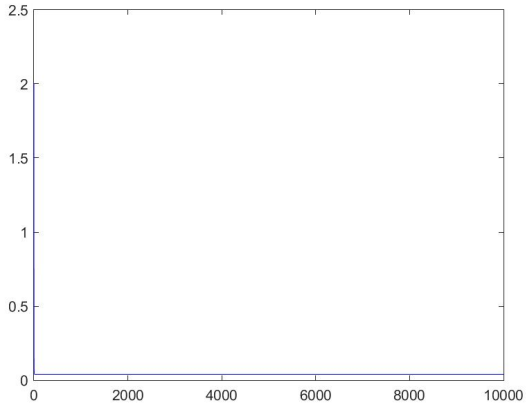


**Figure 7:** Updated error covariance for the velocity state, $P_{22}(k|k)$.

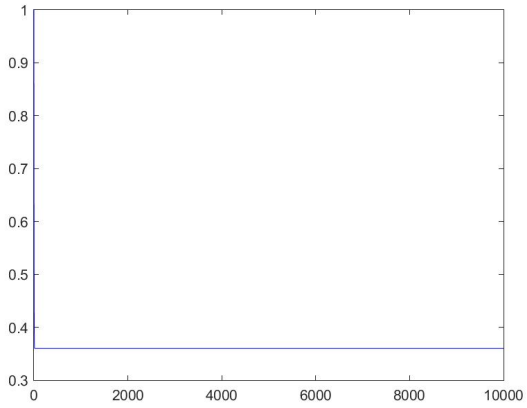


**Figure 8:** First component of the Kalman Gain, $K_{f,1}(k)$.

As shown in Graphs 6 and 7, the updated error covariance elements, $P_{11}(k|k)$ (position variance) and $P_{22}(k|k)$ (velocity variance), converge rapidly from their high initial uncertainty to a stable, steady-state solution, 0.36 and 0.04 respectively. the relationship $\bar{P}_{11} \gg \bar{P}_{22}$ (Position Uncertainty being much higher than Velocity Uncertainty) is a characteristic result of the Constant Velocity model under low process noise: the filter achieves high confidence in its velocity estimate $\hat{x}_2(k|k)$, but the positional uncertainty remains higher because the dominant measurement noise directly affects only the position measurement $y(k)$, and the velocity error is magnified by the sampling period $T = 1$ in the position prediction.

As for Kalman Gain vector $\mathbf{K}_f(k)$, it also converges to a steady-state value with components $\bar{K}_{f1} \approx 0.37$ and $\bar{K}_{f2} \approx 0.08$. The value $\bar{K}_{f1} \approx 0.37$ is the correction factor for the position estimate, which is notably higher than the $1D$ case, as the $2D$ model provides a better prediction baseline for the position. Crucially, the non-zero convergence of the

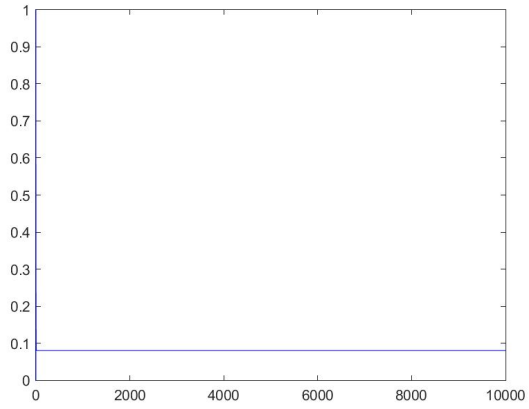


**Figure 9:** Second component of the Kalman Gain, $K_{f,2}(k)$.

second element, $\bar{K}_{f2} \approx 0.08$, confirms the filter's ability to estimate the **unmeasured velocity state**: approximately 8% of the position measurement innovation is used to correct the velocity estimate $\hat{x}_2(k|k)$ by exploiting the cross-correlation between position and velocity errors.

Finally, the calculated performance metrics demonstrate optimal filter performance: the biases (0.0028 and 0.0031) are negligible, confirming the filter provides an **unbiased** estimate, and the calculated variances (0.3618 and 0.1315) are in close agreement with the steady-state error covariance elements $\bar{P}_{11}$ and $\bar{P}_{22}$. This validates its status as an **optimal linear estimator** that minimizes the estimation uncertainty under the given noise conditions.and the calculated variances (0.3618 and 0.1315) are in close agreement with the steady-state error covariance elements $\bar{P}_{11}$ and $\bar{P}_{22}$. **This close alignment between the long-term empirical variances and the theoretical covariance $\mathbf{P}(k|k)$ illustrates the core principle of the Kalman filter: that $\mathbf{P}(k|k)$ accurately models the estimation error uncertainty.** This ultimately validates its status as an **optimal linear estimator** that minimizes the estimation uncertainty under the given noise conditions.

# 3 Part 3

## 3.1 Model and Parameter Setup

The problem involves tracking a target in **circular motion**, which is an inherently nonlinear dynamic system. As the standard Kalman Filter requires a linear model, a critical first step is to establish a suitable **linear approximation** to represent the dynamics.

### 3.1.1 Linear Approximation of Circular Motion

Unlike Part 1 and Part 2, the motion model must be constructed from first principles. Overall, circular motion is modeled as a sequence of discrete rotations with constant angular. A two-dimensional state $\mathbf{x}(k) = [x_1(k), x_2(k)]^T$ (representing the target's position in the $y-z$ plane) is used.
The true non-linear model, which is a function of the angular velocity $\omega$, is approximated by a sequence of linear rotations. By observing the overall trajectory suggested by the $N = 25$ noisy measurements $\mathbf{y}(k)$—which approximate one full circle ($2\pi$ radians)—the constant angular change $\theta$ per time step $k$ for the underlying motion model is inferred and calculated:

$$\theta \approx \frac{2\pi}{N} \approx \frac{2\pi}{24} = \frac{\pi}{12} \quad \text{(Angle per step)}$$

This allows us to construct the linear State Transition Matrix $\mathbf{A}$ as a rotation matrix:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{w}(k) \tag{17}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{v}(k) \tag{18}$$

where the matrices and parameters are:

- State Transition Matrix: $\mathbf{A} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$, with the estimated $\theta = \pi/12$.
- Measurement Matrix: $\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (Direct measurement of both position components).

The initial state is $\mathbf{x}(0) = [0, 0]^T$, as we don't want to rely on extra infomation. The Kalman filter is initialized with a large uncertainty $\mathbf{P}(0|-1) = 10^5\mathbf{I}$.

### 3.1.2 Process Noise Covariance R Design and Justification:

We **design** the Kalman filter by specifying the process noise standard deviation $\sigma_w$ and the measurement noise standard deviation $\sigma_v$:

- We set the measurement noise standard deviation to $\sigma_{\mathbf{v}} = \mathbf{1}$, as per the problem's noise characteristic, reflecting highly corrupted measurements.
- We set the process noise standard deviation to $\sigma_{\mathbf{w}} = \mathbf{0.1}$, consistent with the design in Part 1 and Part 2, which grants the filter a **high but not absolute confidence** in the motion model. This non-zero value is crucial for robustness against the inevitable non-linearity and un-modeled disturbances of circular motion.

The corresponding process and measurement noise covariance matrices are:

$$\mathbf{R}_1 = \sigma_w^2\mathbf{I} = \begin{bmatrix} \mathbf{0.01} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.01} \end{bmatrix} \quad \text{and} \quad \mathbf{R}_2 = \sigma_v^2\mathbf{I} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

## 3.2 Kalman Filter Core Equations and Analytical Results

**Table 1:** Key Scalar Elements of Kalman Filter Matrices over Time ($P_{11}$ in $\mathbf{P}$, $K_{f,1}$ in $\mathbf{K}_f$)

| $k$ | $P_{11}(k+1\|k)$ | $P_{11}(k\|k)$ | $K_{f,1}(k)$ | $K_{11}(k)$ **in** $\mathbf{K}(k)$ | $K_{12}(k)$ **in** $\mathbf{K}(k)$ |
|---|---|---|---|---|---|
| 1 | 100000 | 1.0000 | 1.0000 | 0.9659 | −0.2588 |
| 2 | 1.0100 | 0.5025 | 0.5025 | 0.4854 | −0.1301 |
| 3 | 0.5125 | 0.3388 | 0.3388 | 0.3273 | −0.0877 |
| 4 | 0.3488 | 0.2586 | 0.2586 | 0.2498 | −0.0669 |
| 5 | 0.2686 | 0.2117 | 0.2117 | 0.2045 | −0.0548 |
| 10 | 0.1448 | 0.1265 | 0.1265 | 0.1222 | −0.0327 |
| 15 | 0.1181 | 0.1056 | 0.1056 | 0.1020 | −0.0273 |
| 20 | 0.1097 | 0.0989 | 0.0989 | 0.0955 | −0.0256 |
| **25** | **0.1068** | **0.0965** | **0.0965** | **0.0932** | **-0.0250** |
| **Steady** | $\approx 0.10512$ | $\approx 0.09511$ | $\approx 0.09511$ | $\approx 0.0917$ | $\approx -0.0246$ |

The Kalman filter matrices for any step $k$ are given by the following equations(already shown in (3)-(6)):

$$\mathbf{K}_f(k) = \mathbf{P}(k|k-1)\mathbf{C}^T(\mathbf{C}\mathbf{P}(k|k-1)\mathbf{C}^T + \mathbf{R}_2)^{-1}$$

$$\mathbf{K}(k) = (\mathbf{A}\mathbf{P}(k|k-1)\mathbf{C}^T)(\mathbf{C}\mathbf{P}(k|k-1)\mathbf{C}^T + \mathbf{R}_2)^{-1}$$

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1)$$

$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1$$

Table 1 summarizes the evolution of the key scalar quantities in the Kalman filter. The prediction covariance $P_{11}(k+1|k)$ and updated covariance $P_{11}(k|k)$ contract rapidly from the large initial value $10^5$ once the first measurements are incorporated. By $k = 3$–5, both have already dropped by several orders of magnitude, and by $k \approx 10$ they are essentially indistinguishable from their steady-state levels.

The Kalman gain exhibits the same convergence behavior. The initial value $K_{f,1}(1) = 1$ reflects full reliance on the first measurement; thereafter the gain decreases monotonically as the filter transitions toward model-dominated prediction.

Overall, the table demonstrates that all filter quantities reach steady-state by $k \approx 10$, which is consistent with the contraction properties of rotation-based state transitions under bounded noise. Solving the steady-state Riccati equation yields:

$$\bar{\mathbf{P}} = 0.09511\,\mathbf{I}, \qquad \bar{\mathbf{K}}_f = 0.09511\,\mathbf{I},$$

matching the values observed numerically.

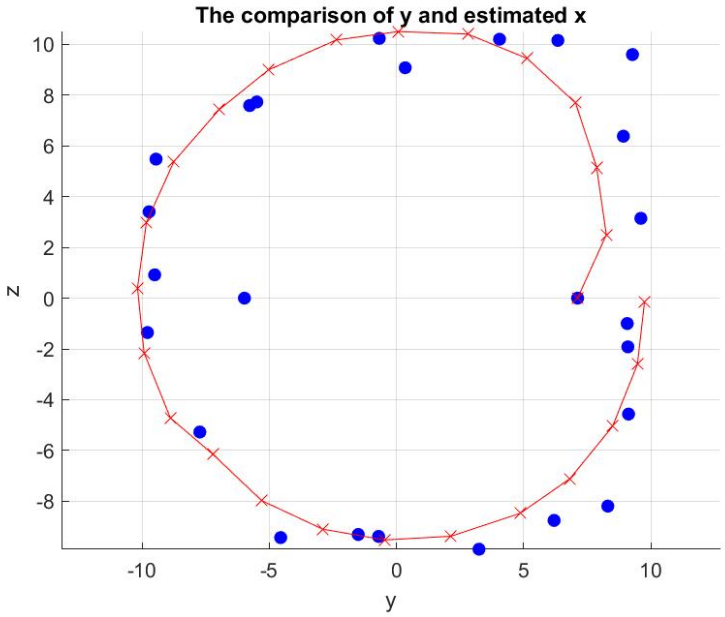


**Figure 10:** The comparison of position measurements $\mathbf{y}(k)$ (blue dots) and Kalman filter estimates $\hat{\mathbf{x}}(k|k)$ (red $\times$ connected by solid-line) in the $y-z$ plane.

### 3.3 Tracking Results and Analysis

The filter achieves significant noise suppression, as visualized in Figure 10. The raw measurements (blue dots) are highly scattered, but the filtered trajectory (red line) is smooth and accurately approximates the circular path. This confirms that the linear rotation model, combined with the low process noise design, effectively captures the underlying dynamics and mitigates the measurement noise.

## A Matlab Code

### A.1 part1

```
N = 10000; % sample
sigma_w = 0.1;
sigma_v = 1;
% true plant:
A = 1;
B = 0;
C = 1;
R1 = sigma_w^2; % covariance of process noise
R2 = sigma_v^2; % covariance of measurement noise
% Pm(:, k) = P(N|N-1)
```

```
Pm(:, 1) = 1e5 * eye(1); % P(0|-1) = 100000 P(1|0) = 100000
% P: covariance of state(uncertainty of an estimate)
% Kalman Gain:
for k = 1: N
    Kf(:, k) = Pm(:, k) * C' * (C * Pm(:, k) * C' + R2)^(-1); % Kf(:, k) = Kf(k)  equation 1.52
    K(:, k) = (A*Pm(:,k)*C')*(C*Pm(:,k)*C'+R2)^(-1); % K(:, k) = K(k)  equation 1.53
    P(:, k) = Pm(:, k) - (Pm(:, k)*C')*(C*Pm(:, k)*C'+R2)^(-1)*C*Pm(:, k); % P(:, k) = P(N|N)  equation 1.56
    Pm(:, k+1) = A*Pm(:, k)*A' + R1 - K(:,k)*(C*Pm(:, k)*C'+R2)*K(:,k)'; % Pm(:, k+1) = P(N+1|N)  equation 1.57
end
% Initialization:
w = random('normal', 0, sigma_w, 1, N);
v = random('normal', 0, sigma_v, 1, N);
xhm(:,1) = [0]'; % xhm(:, 1) = x(0|-1)
% x(:, k) = x(N|N)
x(:, 1) = [5]'; % x(:, 1) = x(0)
for k = 1: N
    x(:, k+1) = A*x(:, k) + w(k); %  x true state
    y(k) = C*x(:,k)+v(k); % measurement
    % Kalman filter:
    xh(:, k) = xhm(:, k) + Kf(:, k) * (y(k) - C * xhm(:, k)); % xh(:, k) = x^hat(k|k)
    xhm(:, k+1) = A*xhm(:, k) + K(:,k)*(y(k) - C*xhm(:, k)); % xhm(:, k+1) = x^hat(k+1|k)
end

% calculate bias:
sum_bias = 0;
for i=1: N
    sum_bias = sum_bias + (x(1, i) - xh(1, i));
end
bias = sum_bias / N;

% calculate variance:
sum_variance = 0;
for i=1: N
    sum_variance = sum_variance + (x(1, i) - xh(1, i))^2;
end
variance = sum_variance / N;

%% 2. plot1
figure('Name','Graph1: x(k) vs x(k|k)');
```

```
plot(1:N, x(1:end-1), 'k-', 'LineWidth', 1.2);    % x(k)
hold on;
plot(1:N, xh,  'r--','LineWidth', 1.);    % x(k|k)
grid on;
xlabel('k');
ylabel('value');
legend('x(k)','x(k|k)');
title('The comparison of x(k) and x_{hat}(k|k)');
saveas(gcf, '1-1.jpg');

%% plot P(k|k) and K_f(k)
figure
plot(1:N, P, 'r-', 'LineWidth', 1.2)
saveas(gcf, '1-2.jpg')

figure
plot(1:N, Kf, 'b--', 'LineWidth', 1.2)
saveas(gcf, '1-3.jpg')

%%
disp(bias)
disp(variance)
```

## A.2 part2

```
N = 10000; % sample
T= 1;
A = [1 T; 0 1];
C = [1 0];
sigma_w = 0.1;
sigma_v = 1;
R1 = [T^4/4 T^3/2; T^3/2 T^2] * sigma_w^2;
R2 = sigma_v^2;
Pm(:,:,1) = 1e5*eye(2); %Pm(:,:,k) = P(N|N-1)
% kalman gain and estimate uncertainty:
for k = 1: N
    Kf(:, k) = Pm(:, :, k) * C' * (C * Pm(:, :, k) * C' + R2) ^ (-1);
    K(:, k) = A * Kf(:, k);
    P(:, :, k) = Pm(:, :, k) - Pm(:, :, k) * C' * (C * Pm(:, :, k) * C' + R2)
    ↪  ^ (-1) * C * Pm(:, :, k);
    Pm(:, :, k+1) = A * Pm(:, :, k)*A' - K(:, k)*(C*Pm(:, :,k)*C' + R2)*K(:,
    ↪  k)' + R1;
end
w = random('normal', 0, sigma_w, 1, N);
v = random('normal', 0, sigma_v, 1, N);
```

```
x(:, 1) = [0; 30];
xhm(:, 1) = [0; 0];
for k = 1: N
    x(:, k+1) = A*x(:, k) + [T^2/2; T] * w(:, k);
    y(k) = C * x(:, k) + v(:, k);
    xh(:, k) = xhm(:, k) + Kf(:, k)*(y(k) - C*xhm(:, k));
    xhm(:, k+1) = A*xhm(:, k) + K(:, k)*(y(:, k) - C*xhm(:, k));
end
% calculate biases:
%x1:
sum_bias_x1 = 0;
for i=1: N
    sum_bias_x1 = sum_bias_x1 + (x(1, i) - xh(1, i));
end
bias_x1 = sum_bias_x1 / N;

%x2:
sum_bias_x2 = 0;
for i=1: N
    sum_bias_x2 = sum_bias_x2 + (x(2, i) - xh(2, i));
end
bias_x2 = sum_bias_x2 / N;

% calculate variance:
sum_variance_x1 = 0;
for i=1: N
    sum_variance_x1 = sum_variance_x1 + (x(1, i) - xh(1, i))^2;
end
variance_x1 = sum_variance_x1 / N;

%x2:
sum_variance_x2 = 0;
for i=1: N
    sum_variance_x2 = sum_variance_x2 + (x(2, i) - xh(2, i))^2;
end
variance_x2 = sum_variance_x2 / N;

%%
figure
plot(1:N,x(1,1:N),'k-',1:N,xh(1,:),'r--')
legend('x1(k)','x1(k|k)')
saveas(gcf,'2-1.jpg')

figure
plot(1:N,x(2,1:N),'k-',1:N,xh(2,:),'r--')
legend('x2(k)','x2(k|k)')
```

```
saveas(gcf,'2-2.jpg')

figure
plot(1:N,squeeze(P(1,1,:)),'b')
saveas(gcf,'2-3.jpg')

figure
plot(2:N,squeeze(P(2,2,2:N)),'b')
saveas(gcf,'2-41.jpg')

figure
plot(1:N,Kf(1,:),'b')
saveas(gcf,'2-5.jpg')

figure
plot(1:N,Kf(2,:),'b')
saveas(gcf,'2-6.jpg')

%%
disp(bias_x1)
disp(bias_x2)
disp(variance_x1)
disp(variance_x2)
```

## A.3 part3

```
%% data init
N = 25;
theta = pi / 12;
A = [cos(theta) -sin(theta); sin(theta) cos(theta)];

R1 = [0.01 0; 0 0.01]; % process uncertainty
R2 = [1 0; 0 1];  % measurement uncertainty
C = eye(2);

y(1,1) = 7.1165;
y(1,2) = 9.6022;
y(1,3)=8.9144;
y(1,4)=9.2717;
y(1,5)=6.3400;
y(1,6)=4.0484;
y(1,7)=0.3411;
y(1,8)=-0.6784;
y(1,9)=-5.7726;
y(1,10)=-5.4925;
```

```
y(1,11)=-9.4523;
y(1,12)=-9.7232;
y(1,13)=-9.5054;
y(1,14)=-9.7908;
y(1,15)=-7.7300;
y(1,16)=-5.9779;
y(1,17)=-4.5535;
y(1,18)=-1.5042;
y(1,19)=-0.7044;
y(1,20)=3.2406;
y(1,21)=8.3029;
y(1,22)=6.1925;
y(1,23)=9.1178;
y(1,24)=9.0904;
y(1,25) = 9.0662;
y(2,1) = 0.000;
y(2,2) = 3.1398;
y(2,3)=6.3739;
y(2,4)=9.5877;
y(2,5)=10.1450;
y(2,6)=10.1919;
y(2,7)=9.0683;
y(2,8)=10.2254;
y(2,9)=7.5799;
y(2,10)=7.7231;
y(2,11)=5.4721;
y(2,12)=3.3990;
y(2,13)=0.9172;
y(2,14)=-1.3551;
y(2,15)=-5.2708;
y(2,16)-9.7011;
y(2,17)=-9.4256;
y(2,18)=-9.3053;
y(2,19)=-9.3815;
y(2,20)=-9.8822;
y(2,21)=-8.1876;
y(2,22)=-8.7501;
y(2,23)=-4.5653;
y(2,24)=-1.9179;
y(2,25)=-1.0000;

Pm(:,:,1) = 1e5*eye(2); %P(N|N-1) %%
xhm(:, 1) = [0; 0];
%% pre compute: kalman gain and estimate uncertainty
for k=1: N
    Kf(:,:, k) = Pm(:, :, k) * C' * (C * Pm(:, :, k) * C' + R2) ^ (-1);
```

```
66      K(:,:, k) = A * Kf(:,:, k);
67      P(:, :, k) = Pm(:, :, k) - Pm(:, :, k) * C' * (C * Pm(:, :, k) * C' + R2)
        ↪  ^ (-1) * C * Pm(:, :, k);
68      Pm(:, :, k+1) = A * Pm(:, :, k)*A' - K(:,:, k)*(C*Pm(:, :,k)*C' +
        ↪  R2)*K(:,:, k)' + R1;
69  end
70
71  %% estimation
72  for k=1: N
73      xh(:, k) = xhm(:, k) + Kf(:,:, k)*(y(:,k)-C*xhm(:,k));
74      xhm(:, k+1) = A*xhm(:, k) + K(:,:, k)*(y(:, k)-C*xhm(:, k));
75  end
76  %% plot
77  figure;
78  hold on;
79  axis equal;
80  plot(y(1,:), y(2,:), 'bo', 'MarkerFaceColor', 'b');
81
82  plot(xh(1,:), xh(2,:), 'rx-', 'LineStyle', '-', 'Color', 'r', 'MarkerSize',
    ↪  8);
83  xlabel('y'); ylabel('z');
84  title('The comparison of y and estimated x');
85  grid on;
86  saveas(gcf, '3-1.jpg');
87
```