Project for EE5103 Computer Control Systems

Submitted by
Jia Yansong A0263119H

2022.11.13

Session 2022/2023

Abstract

This paper is the project of EE5103, and it contains three parts using Kalman Filter algorithm to compute and simulate three different discrete time state-space models. In the third part, rotation matrix is used to construct true model for the system and Kalman Filter is used to estimate the states and compare with the ground truth trajectory.

Content

1. Introduction

Consider the discrete time model:

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{1}$$

$$y(k) = Cx(k) + v(k) \tag{2}$$

Where $w(k)$ and $v(k)$ are zero-mean independent Gaussian noise with covariance matrix R_1 and R_2 respectively. The Kalman Filter is given as:

$$K_f(k) = P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1} \tag{3}$$

$$K(k) = (AP(k|k-1)C^T)(CP(k|k-1)C^T + R_2)^{-1} \tag{4}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_f(k)(y(k) - C\hat{x}(k|k-1) \tag{5}$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{6}$$

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1) \tag{7}$$

$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1 \tag{8}$$

Because of Least-Square Estimate, it is proved that Kalman Filter is the best estimation method because the error between estimate values and true values are the smallest.

Equation (3) is to compute Kalman Gain K_f, equation (5) is to update the state value calculated by Kalman Gain with measurements and estimate values, equation (6) is to estimate the next step's states' values at present step, equation (7) is to update the covariance of states (uncertainty of an estimate), and finally equation (8) is to estimate next step's uncertainty of the estimate at present step. And in the third part, rotation matrix is also used to model the ground truth system.

MATLAB code *part1.m*, *part2.m*, *part3.m* are part 1, part 2, part 3 code respectively, they are used to compute the states and requirements in each part, and they are all in appendix section.

2. Part 1

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

$$x(k+1) = x(k) + w(k) \tag{9}$$

$$y(k) = x(k) + v(k) \tag{10}$$

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 initial condition $x(0) = 5$.

First set the initial covariance of state $P(0|-1) = 100000$.

$$k = 0, 1, \ldots, N \; where \; N = 10000$$

Use K_f matrix store Kalman Gain value, x matrix stores values $x(k|k)$, P matrix stores values $P(k|k)$, x_m matrix stores values $x(k+1|k)$, P_m matrix stores values $P(k+1|k)$.

The initialization of P_m is set to 10000, $P_m(:,1) = 100000$. Use equation (3), (4), (7), (8) to calculate Kalman Gain and update the estimate uncertainty and extrapolate uncertainty:

$$K_f(k) = P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1} \tag{3}$$

$$K(k) = (AP(k|k-1)C^T)(CP(k|k-1)C^T + R_2)^{-1} \tag{4}$$

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1) \tag{7}$$

$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1 \tag{8}$$

Then use equation (5) and (6) to update estimate with measurement and extrapolate the state:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_f(k)(y(k) - C\hat{x}(k|k-1) \tag{5}$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{6}$$

True state and measurement are:

$$x(k+1) = x(k) + w(k) \tag{9}$$

$$y(k) = x(k) + v(k) \tag{10}$$

Because $\sigma_w = 0.1$ and $\sigma_v = 1$, w and v can be derived using MATLAB command: $random$ to create N zero-mean independent Gaussian random variables with σ_w and σ_v.

Use MATLAB code in *part1.m* to calculate bias:

$$Bias = \frac{1}{N+1}\sum_{k=0}^{N}\big(x(k) - \hat{x}(k|k)\big) = -4.6215 \times 10^{-4} \tag{11}$$

Calculate variance:

$$Variance = \frac{1}{N+1}\sum_{k=0}^{N}\big(x(k) - \hat{x}(k|k)\big)^2 = 0.0923 \tag{12}$$

Graph1: $x(k)\ (solid-line), \hat{x}(k|k)\ (dotted-line)$

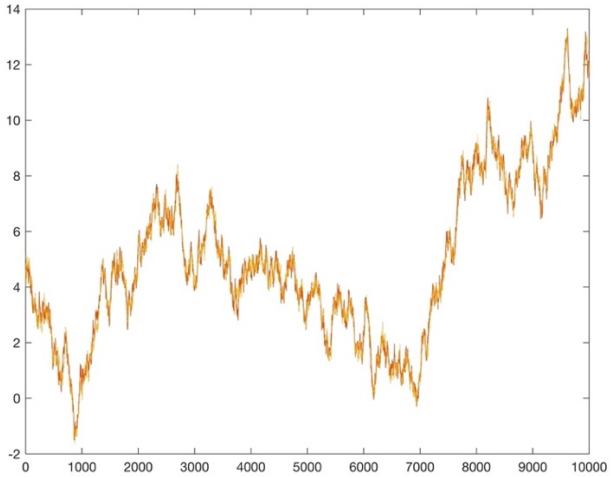

Figure 1 x(k)(solid-line) and estimated x(k)(dotted-line)

Graph2: $P(k|k)$

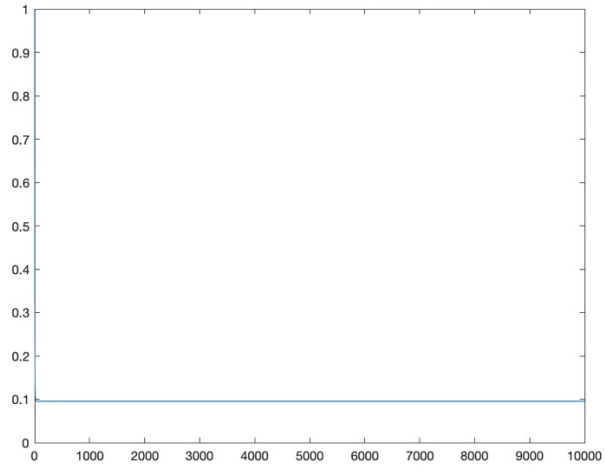

Figure 2 P(k|k)

Graph3: $K_f(k)$

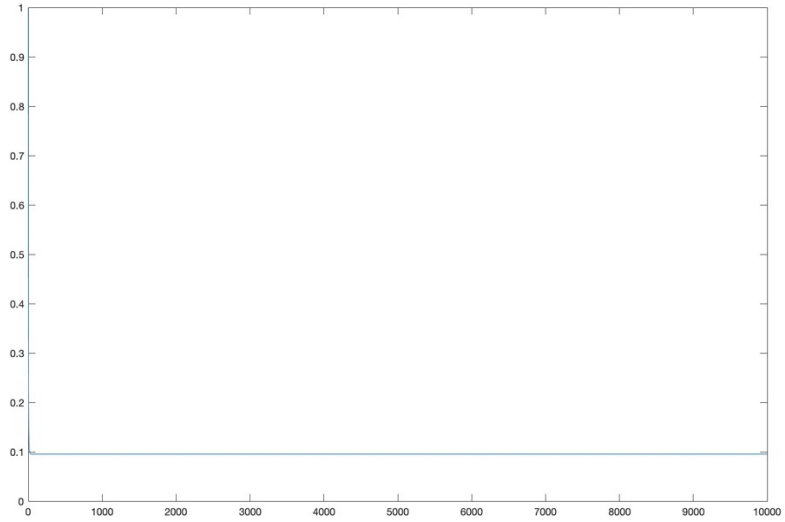

Figure 3 Kf(k)

3. Part 2:

$x_1(k)$ represents the true state position and $x_2(k)$ represents the velocity of a moving target.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} w(k) \tag{13}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v(k) \tag{14}$$

Where $w(k)$ and $v(k)$ are zero-mean independent Gaussian random variables with standard derivations $\sigma_w = 0.1$ and $\sigma_v = 1$ respectively. The sampling period is $T = 1$ and the initial conditions are $x_1(0) = 0$ and $x_2(0) = 30$.

Use equation (3), (4), (7), (8) to calculate Kalman gain and estimate uncertainty by these equations:

$$K_f(k) = P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1} \tag{3}$$

$$K(k) = (AP(k|k-1)C^T)(CP(k|k-1)C^T + R_2)^{-1} \tag{4}$$

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1) \tag{7}$$

$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1 \tag{8}$$

Then, Kalman gain's values in different samples are stored in K_f matrix, current states' uncertainties' values are stored in P matrix, and extrapolated states' uncertainties' values are stored in P_m matrix.

Then use equation (9), (10) to calculate true states and use equation (5) and (6) to estimated states using Kalman Filter:

$$x(k+1) = x(k) + w(k) \tag{9}$$

$$y(k) = x(k) + v(k) \tag{10}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_f(k)(y(k) - C\hat{x}(k|k-1) \tag{5}$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{6}$$

True states are stored in x matrix, measurement values are stored in y matrix, estimated states are stored in xh matrix and estimated extrapolated states are stored in xhm matrix.

Graph 4, graph 5, graph 6, graph 7, graph 8 and graph 9 are based on values in matrix x, xh, P, K_f.

Graph 4: $x_1(k)(solid-line), \hat{x}(k|k)(dotted-line)$:

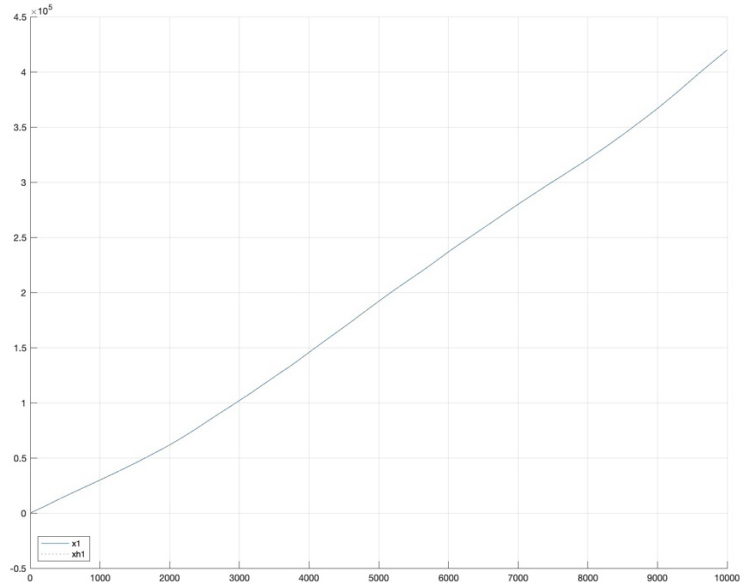

Figure 4 x1(k) and x estimate

Graph 5: $x_2(k)(solid-line), \hat{x}(k|k)(dotted-line)$:

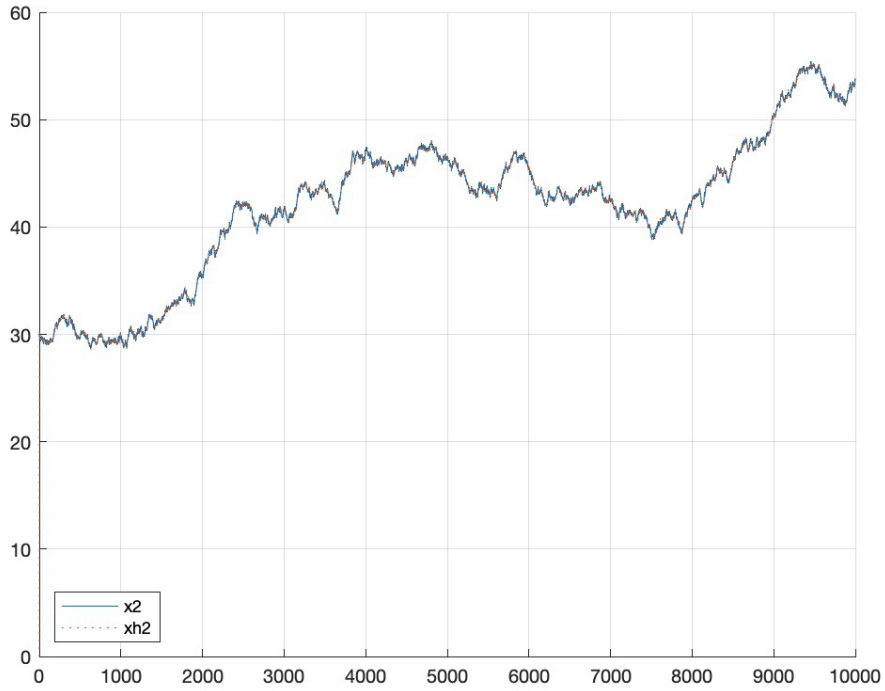

Figure 5 x2(k) and estimate x(k|k)

Graph 6: $P_{11}(k|k)$:

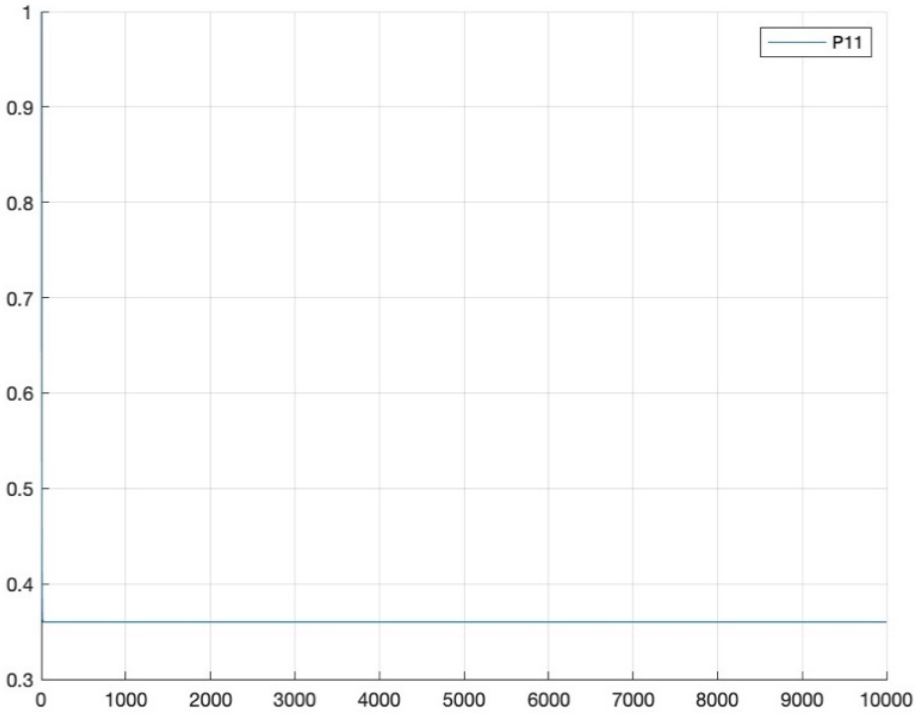

Figure 6 P11(k|k)

Graph 7: $P_{22}(k|k)$:

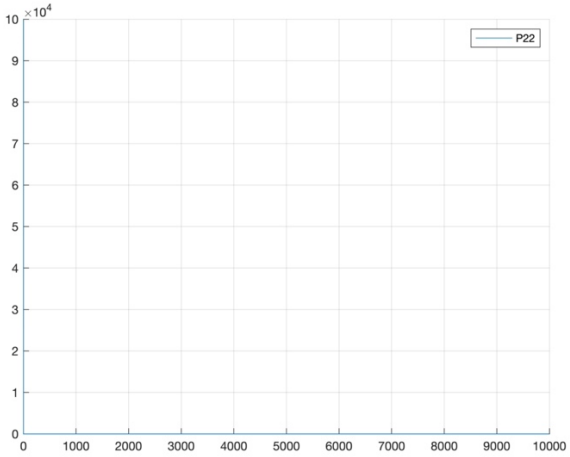

Figure 7 P22(k/k)

Graph 8: $K_{f1}(k)$:

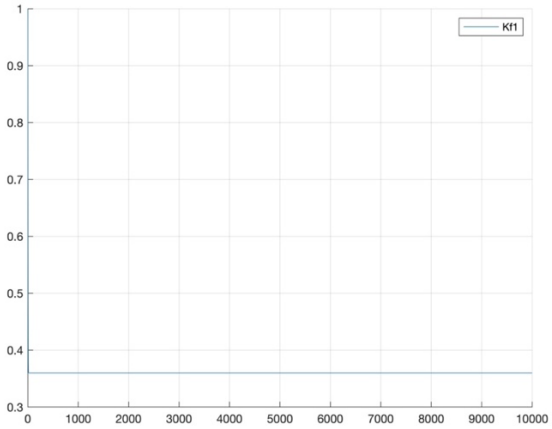

Figure 8 Kf1(k)

Graph 9: $K_{f2}(k)$:

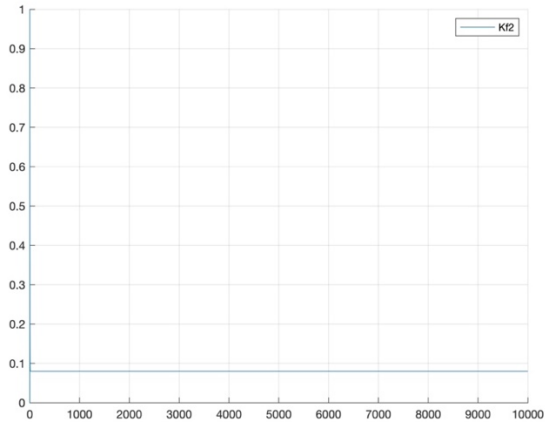

Figure 9 Kf2(k)

Use MATLAB code in *part2.m* to calculate:

Calculate the biases:

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(x_1(k)-\hat{x}_1(k|k)\right)=0.0296 \tag{15}$$

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(x_2(k)-\hat{x}_2(k|k)\right)=0.0124 \tag{16}$$

Calculate the variances:

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(x_1(k)-\hat{x}_1(k|k)\right)^2=0.3424 \tag{17}$$

$$\frac{1}{N}+1\sum_{k=0}^{N}\left(x_2(k)-\hat{x}_2(k|k)\right)^2=0.1288 \tag{18}$$

4. Part 3:

From the trajectory of the target, it is obviously that the target is moving around an origin point, and its trajectory is a circle whose radius is 10, and origin is $(0,0)$. It is given that the true state positions on the yz-plane $\left(10\cos\frac{1}{12}k\pi, 10\sin\frac{1}{12}k\pi\right), k = 0,1,\dots,N \text{ where } N=24.$

Rotation matrix is normally used in rotating a point around a center point, the angle difference between two samples is counter clockwise $\frac{\pi}{12}$degree, so the rotation matrix is $:\begin{bmatrix}\cos\frac{\pi}{12} & -\sin\frac{\pi}{12}\\ \sin\frac{\pi}{12} & \cos\frac{\pi}{12}\end{bmatrix}$, and motion model is $\begin{bmatrix}y(k+1)\\ z(k+1)\end{bmatrix}=\begin{bmatrix}\cos\frac{\pi}{12} & -\sin\frac{\pi}{12}\\ \sin\frac{\pi}{12} & \cos\frac{\pi}{12}\end{bmatrix}\begin{bmatrix}y(k)\\ z(k)\end{bmatrix}, k = 0,1,\dots,N \text{ where } N=24.$

In order to avoid causing ambiguity with y, h represents $y(k)$ and $z(k)$ measurements, and the state space model is designed:

$$\begin{bmatrix}y(k+1)\\ z(k+1)\end{bmatrix}=\begin{bmatrix}\cos\frac{\pi}{12} & -\sin\frac{\pi}{12}\\ \sin\frac{\pi}{12} & \cos\frac{\pi}{12}\end{bmatrix}\begin{bmatrix}y(k)\\ z(k)\end{bmatrix}+w(k) \tag{19}$$

$$h(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y(k) \\ z(k) \end{bmatrix} + v(k) \tag{20}$$

Initialization:

Because the true state motion model of this target is clear, set process uncertainty is zero, $w(k) = 0, R_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. The measurements are noisy, so set measurement uncertainty higher, $v(k) = 1, R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Because initial states are quite uncertain, $y(k)\ and\ z(k)$ are independent, $cov(y, z) = 0\ and\ cov(z, y) = 0$, set initial covariance of states $P(0|-1) = \begin{bmatrix} 10000 & 0 \\ 0 & 10000 \end{bmatrix}$.

Use equation (3) and (4) to calculate Kalman gain and use equation (7) and (8) to estimate uncertainty by these equations:

$$K_f(k) = P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1} \tag{3}$$

$$K(k) = (AP(k|k-1)C^T)(CP(k|k-1)C^T + R_2)^{-1} \tag{4}$$

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T(CP(k|k-1)C^T + R_2)^{-1}CP(k|k-1) \tag{7}$$

$$P(k+1|k) = AP(k|k-1)A^T - K(k)(CP(k|k-1)C^T + R_2)K^T(k) + R_1 \tag{8}$$

Then create a matrix y, stores $y\ and\ z$ measurements.

Then use equation (9) and (10) to calculate true states and use equation (5) and (6) to estimated states using Kalman Filter:

$$x(k+1) = x(k) + w(k) \tag{9}$$

$$y(k) = x(k) + v(k) \tag{10}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_f(k)(y(k) - C\hat{x}(k|k-1) \tag{5}$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{6}$$

K_f, P, P_m, x and xh store Kalman Gain in different samples, uncertainty of state estimate, extrapolated next sample's uncertainty of state estimate, true state and estimated state by Kalman Filter values.

Use these matrixes, draw final results: estimates $\left(\hat{x}_y(k|k), \hat{x}_z(k|k)\right)$ using $\times$.

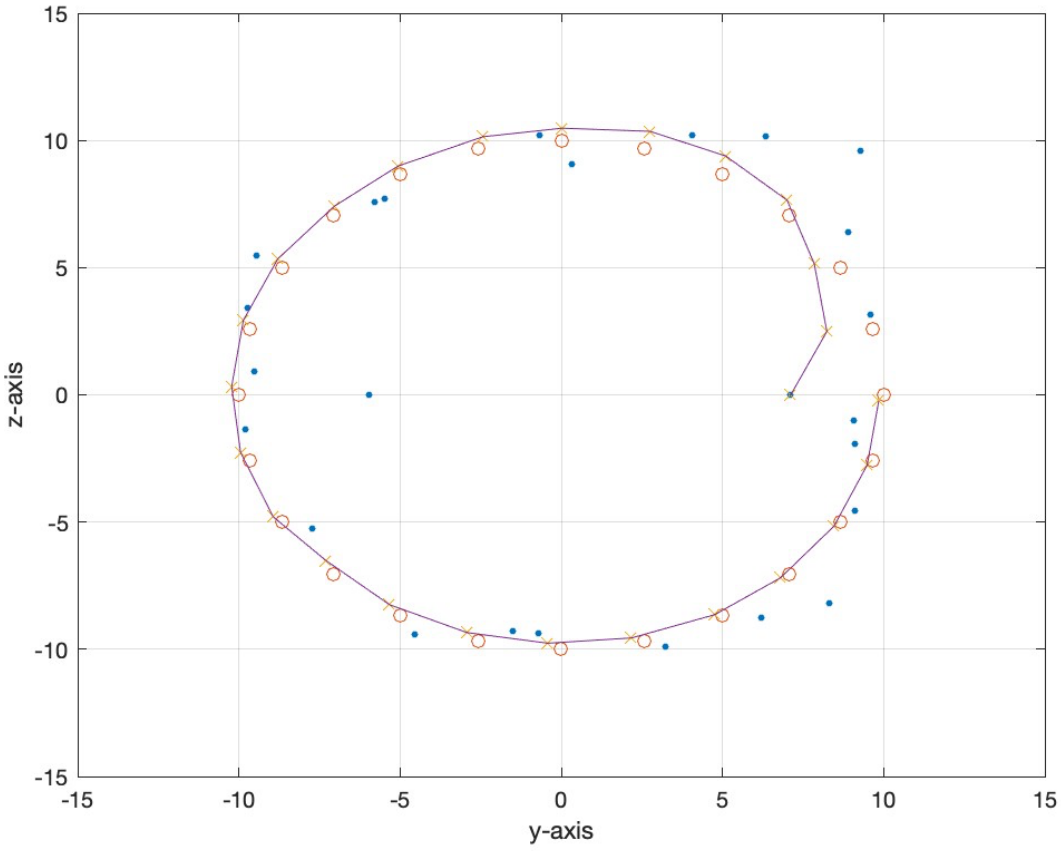

Figure 10 estimate trajectory with groudtruth and measurement

Use MATLAB code in *part3.m* writing loops to calculate:
Calculate the bias:

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(10\cos\frac{2\pi k}{N}-\hat{x}_y(k|k)\right)=0.3470 \tag{21}$$

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(10\sin\frac{2\pi k}{N}-\hat{x}_z(k|k)\right)=-0.2477 \tag{22}$$

Calculate variances:

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(10\cos\frac{2\pi k}{N}-\hat{x}_y(k|k)\right)^2=0.4871 \tag{23}$$

$$\frac{1}{N+1}\sum_{k=0}^{N}\left(10\sin\frac{2\pi k}{N}-\hat{x}_z(k|k)\right)^2=0.1340 \tag{24}$$

5. Conclusion

These three parts are all using Kalman Filter to estimate states of the systems with true groudtruth values and measurements. In the first two parts, the uncertainty of R_1 and R_2 are given, and use Kalman Filter algorithm's equation set to iterate and update every variable in this algorithm and estimate state values. In the third part, because it is clear that the target is truly moving around the origin, so I believe that the process's uncertainty is zero, so I set R_1 equals to zero matrix. And from the final trajectory of estimate states we can see that my decision is right because the measurement is very noisy, and the model is true, the estimate of states are converging to groudtruth values.

Appendix

part1.m:

```
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 = [0]'; % xhm(:, 1) = x(0|-1) or x(1|0)
% x(:, k) = x(N|N)
x(:, 1) = [5]; % x(:, 1) = x(0) or x(1) (matlab's matrix don't have 0 subscript)
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;
```

part2.m

```
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;
```

part3.m

```
N = 25;
theta = pi / 12;
A = [cos(theta) -sin(theta); sin(theta) cos(theta)];
R1 = [0 0; 0 0]; % process uncertainty
R2 = [1 0; 0 1];  % measurement uncertainty
C = eye(2);
Pm(:,:,1) = 1e5*eye(2); %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, 1, 1, N);
v = random('normal', 0, 1, 2, N);
x(:, 1) = [10; 0];
xhm(:, 1) = [10; 0];
% put measurement to y
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;
for k=1: N
   % x(:, k+1) = A*x(:, k) + [1; 1]*w(:, k);
   x(:, k+1) = A*x(:, 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
% biases:
num = 0;
for k = 1: N
   % num = num + 10 * cos(2 * pi * (k-1) / (N-1)) - xh(1, k);
   num = num + x(1,k) - xh(1,k);
end
y_bias = num / N;
num = 0;
for k = 1: N
   num = num + x(2,k) - xh(2,k);
end
z_bias = num / N;
% variance:
num = 0;
for k = 1: N
   num = num + (x(1,k) - xh(1,k))^2;
end
y_variance = num / N;

num = 0;
for k = 1: N
   num = num + (x(2,k) - xh(2,k))^2;
end
z_variance = num / N;
```