Kalman Filter and Extended Kalman Filter

Messiah Abolfazli Esfahani

October 2022

1 Introduction

In this Assignment, the Estate estimation using the Kalman filter and extended Kalman filter is being investigated. The development is done in Python programming language by using numpy and PyGame.PyGame is a Python wrapper for the SDL library, which stands for Simple Direct Media Layer. SDL provides cross-platform access to your system's underlying multimedia hardware components, such as sound, video, mouse, keyboard, and joystick.

2 Kalman Filter

Most modern systems have numerous sensors that estimate hidden (unknown) states based on a series of measurements. For example, a GPS receiver provides location and velocity estimation, where location and velocity are the hidden states and differential time of the satellites' signals' arrival are the measurements.

One of the biggest challenges of tracking and control systems is providing an accurate and precise estimation of the hidden states in the presence of uncertainty. In GPS receivers, the measurement uncertainty depends on many external factors such as thermal noise, atmospheric effects, slight changes in satellite positions, receiver clock precision, and many more.

The Kalman Filter is one of the most important and common estimation algorithms. The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements. Also, the Kalman Filter predicts the future system state based on past estimations. The general overview of state estimation is shown in the equation below:

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n + w_n \tag{1}$$

In the equation 1:
$x_{n+1,n}$ is the predicted system state vector at time step $n+1$.
$x_{n,n}$ is the predicted system state vector at time step n.
u_n is the control variable or input variable.
w_n is the process noise or distrubance.

F is the state transition matrix.
G is a control matrix or input transition matrix.
If we consider the system a 2D system, the state matrix, $\hat{x}_n$, can be shown as follow:

$$\begin{bmatrix} \hat{x} \\ \hat{\dot{x}} \\ \hat{y} \\ \hat{\dot{y}} \end{bmatrix}$$

There exists a measurement as well. That can be computed as follow:

$$Z_n = H\hat{x}_n + v_n \tag{2}$$

By using the equation 2 the actual from sensors arrive. In the equation 2 v_n is called the measurement noise and H is called measurement matrix. Also the state covariance is shown by $\hat{P}_0$ and measurement noise covariance is being shown by R and process noise covariance is shown by Q. state covariance matrix can be computed as follow:

$$P = FPF^T + Q \tag{3}$$

The Kalman-gain is the weight given to the measurements and current-state estimate, and can be "tuned" to achieve a particular performance. With a high gain, the filter places more weight on the most recent measurements, and thus conforms to them more responsivel and can be computed as follow:

$$K = P_{n,n-1}H^T/(HP_{n,n-1}H^T + R) \tag{4}$$

After computing the Kalman gain the position and covariance will be updated accordingly. As a result the new position estimation and state covariance can be computed as follow:

$$\hat{x_k} = \hat{x_k} + K(z - Hx) \tag{5}$$

$$\hat{P_k} = (I - KH)P \tag{6}$$

2.1 Implementation

The implementation is done in Python by using Pygame. At top left corner the covariance matrix is plotted as an ellipse, and blue, green and red rectangles show predicted position, Ground Truth and measured position respectively.

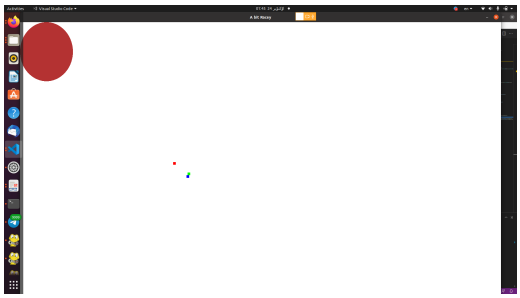

Figure 1: covariance values over iterations

2.2 Question1

In this question the prediction step is 8 times of the measurement step. The Transition matrix, F, is as follow:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The state matrix is $\hat{x}_k = \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix}$ .The control matrix G= $\begin{bmatrix} r\Delta t/2 & r\Delta t/2 \\ r\Delta t/2 & r\Delta t/2 \end{bmatrix}$. Process noise is $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.15 \end{bmatrix}$. The car is supposed to move from left top corner that is the point(0,0) to the right and the state matrix is supposed to be $p = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. The car is moving from left to right 1 meter. It should be mentioned that the car speed is the same over all iterations. The observation matrix is of the form $\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$.
The final formulation is:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \end{bmatrix} + \begin{bmatrix} r\Delta t/2 & r\Delta t/2 \\ r\Delta t/2 & r\Delta t/2 \end{bmatrix} \begin{bmatrix} u_r \\ u_l \end{bmatrix} + w_n \Delta t \tag{7}$$

The most important observation of this experiment is that, by coming the observation the state covariance matrix is decreased and as expected the Kalman filter moves more smooth rather than normal movement. In the Figure 1 and 2 the x and y values of prediction and measurement is plotted.As can be seen, the predictions are moved towards the estimations, whenever the estimation is arrived. Looking deeper into it we can see that over the period of 1 second the covariance increases (refer to Figure 3), and as covariance is increased the prediction moves more towards using the

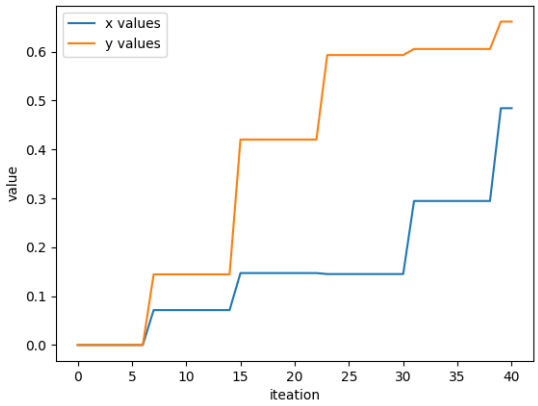

Figure 2: Measurement values with noise over iterations

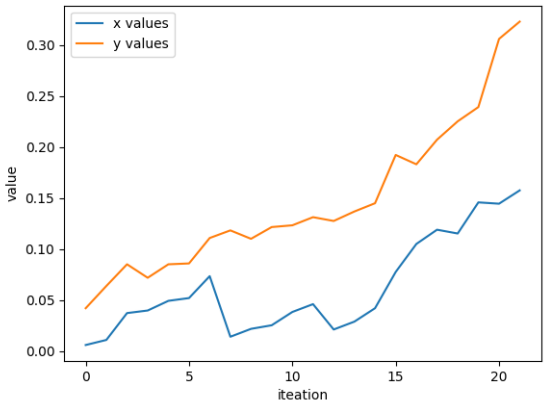

Figure 3: Prediction values over iterations

2.3 Question2

In the question 2, the system is supposed to rotate around the point $M = [10, 10]$ and $L = 0.3$ is the distance between wheels. The velocity in the x and y and angle direction can be computed by using the following equations:

$$\dot{x} = r/2(u_r + u_l)cos(\theta) + w_x \tag{8}$$

$$\dot{x} = r/2(u_r + u_l)sin(\theta) + w_y \tag{9}$$

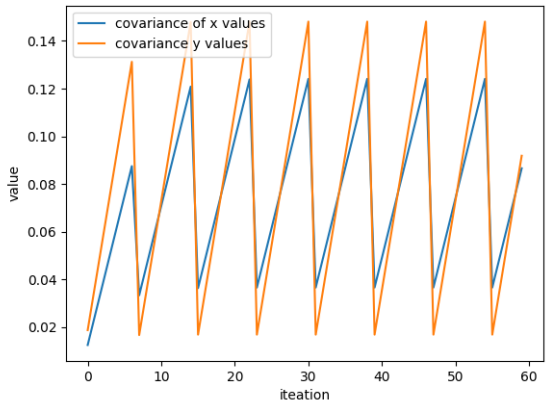

Figure 4: covarriance values over iterations

$$\dot{\theta} = r(u_r - u_l)/L \tag{10}$$

In order get the car rotate along the landmark point, the control signals must be applied to wheels, in a manner that if the distance is being greater, the left wheel signal will be applied only (in a left circular rotation along the point). The rest of formulation is the same. Like previous question, by combining the observation the state covariance is being decreased and estimated positions move more smooth..The overall formulation is similar to 11

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \\ \hat{\theta}_n \end{bmatrix} + \begin{bmatrix} r\Delta t cos(\theta) & 0 \\ r\Delta t sin(\theta) & 0 \\ 0 & \Delta t r/l \end{bmatrix} \begin{bmatrix} u_r \\ u_l \end{bmatrix} + w_n \Delta t \tag{11}$$

The state matrix will be $P_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

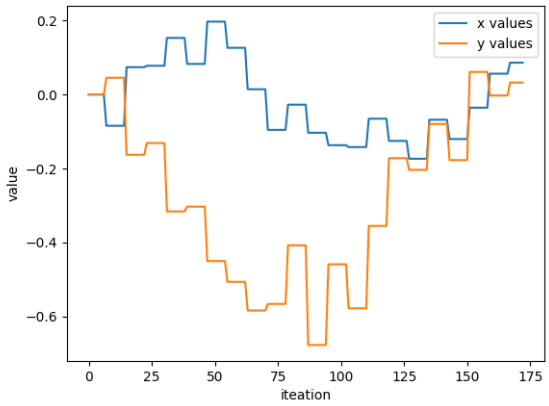

Figure 5: Measurement values with noise over iterations

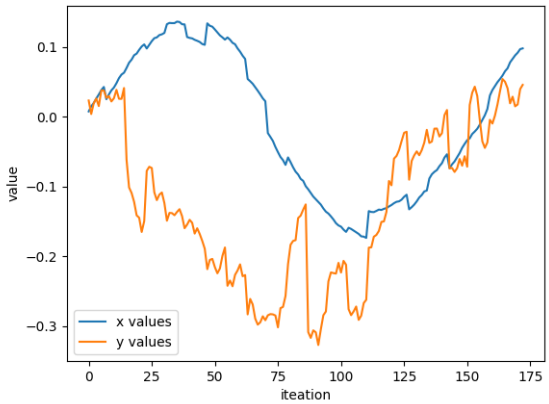

Figure 6: Prediction values over iterations

As can be seen in figure 4,5 and 6, The covariance has increased by coming a measurement and prediction moves smoother rather than measurement values.

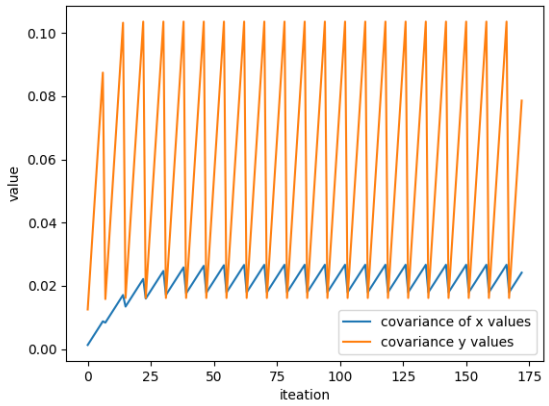

Figure 7: covarriance values over iterations

2.4 Question 3

We can formulate Cartesian coordinate system in polar as $\begin{bmatrix} \sqrt{(p_x - x)^2 + (p_y - y)^2} \\ \arctan(p_y - y)/(p_x - x) - \theta \end{bmatrix}$

where (p_x, p_y) is the landmark and (x,y) are the position of the vehicle in the landmark coordinate. x and y are positions in Lidar system. By converting the System to polar.The H matrix also must be modified and follow the Jacobian of the above transform which can be written as $\begin{bmatrix} \frac{-p_x+x}{\sqrt{x^2+y^2}} & \frac{-p_y+y}{\sqrt{x^2+y^2}} & 0 \\ \frac{y-p_y}{\sqrt{(p_x-x)^2+(p_y-y)^2}} & \frac{x-p_x}{\sqrt{(p_x-x)^2+(p_y-y)^2}} & -1 \end{bmatrix}$.

The R can be defined as below. $R = \begin{bmatrix} \sigma^2_{range} & 0 \\ 0 & \sigma^2_{bearing} \end{bmatrix}$.

The result of this section was not good mainly due to the large noise existing in the r and theta measurements. Based on the simulation, the small noise in r and theta is affecting the system by a lot.

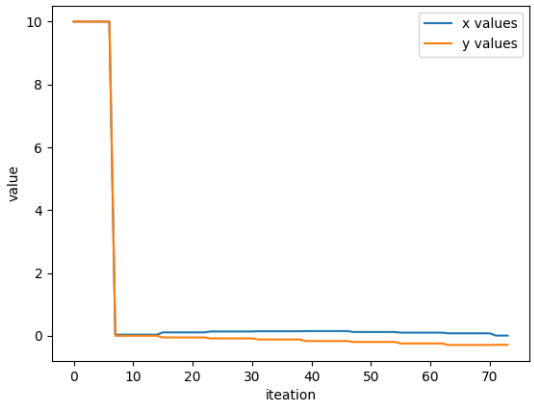

Figure 8: Measurement values with noise over iterations

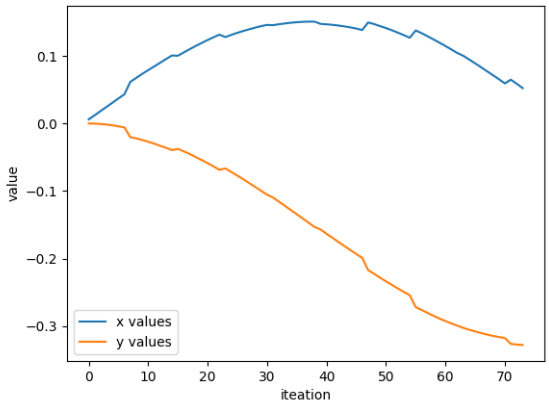

Figure 9: Prediction values over iterations

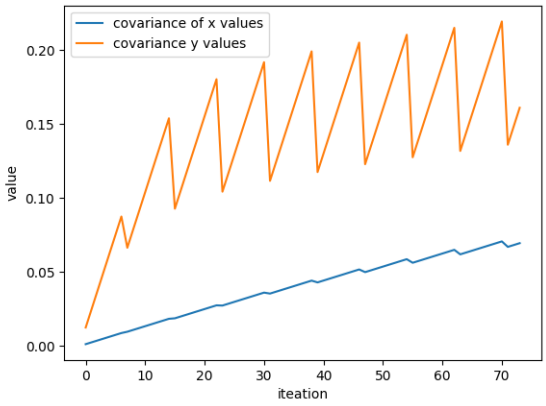

Figure 10: covarriance values over iterations