Lab 2

EE183DA- Design of Robotic Systems

By Victor Rios, Sokchetra Eung, and Cooper Simpson

2/13/2020

Abstract

Within this lab we aimed to create and implement in code a Kalman Filter, whose purpose is to estimate the state of a system with a best guess and certainty on that guess. The system being described is the previously used system from lab 1 (reference to that lab found at end) that is comprised of a two wheel car equipped with various sensors. Since this system is not a linear system, we had to construct an Extended Kalman Filter (EKF) and a portion of this lab was dedicated to deriving this EKF from known system dynamics and measurement equations. The remainder of this lab was dedicated to implementing this code in software and then running various experiments to evaluate the performance of the EKF under various conditions. The performance is gauged by noting the evolution of covariance of the EKF as well as the accuracy of the state estimator by comparing the estimate to the value produced during simulation. After experimentation, it was found that the EKF is in fact a reasonable state estimator, and was able to provide accurate state information despite significant uncertainty in initial position.

System Recap

As a quick reminder, the system, its dynamics and its measurements are restated here. The system has four states and they are as follows

$$x : \textit{x position}$$
$$y : \textit{y position}$$
$$\theta : \textit{orientation of car}$$
$$\theta' : \textit{angular velocity}$$

The system has two inputs which pulse-width modulation signals, one for each wheel, but these are mapped to angular velocities so these are used as the inputs to the system

$$\omega_l : \textit{left wheel angular velocity}$$
$$\omega_r : \textit{right wheel angular velocity}$$

The system produces five outputs and they are as follows

$$d : \textit{distance from rangefinder on front}$$
$$l : \textit{distance from rangefinder on back}$$
$$m_x : \textit{magnetometer reading along x axis}$$
$$m_y : \textit{magnetometer reading along y axis}$$
$$g : \textit{gyro reading around axis}$$

The following are constants used in equations and that will be defined here

$$L : \textit{distance from origin of east wall}$$
$$W : \textit{distance from origin of west wall}$$
$$R_w : \textit{radius of wheel}$$
$$D : \textit{distance from wheel to center of car}$$
$$M : \textit{scalar used for magnetometer reading}$$
$$T : \textit{sampling period}$$

Lastly, there actuator noise w and sensor noise v which can be described as additive white gaussian noise (AWGN). Subscript of w and v tells you which variable this noise effects

$$w_x \ -N(0,\sigma_x{}^2)$$
$$w_y \ -N(0,\sigma_y{}^2)$$
$$w_\theta \ -N(0,\sigma_\theta{}^2)$$
$$w_{\theta'} \ -N(0,\sigma_{\theta'}{}^2)$$
$$v_d \ -N(0,\sigma_d{}^2)$$
$$v_l \ -N(0,\sigma_l{}^2)$$
$$v_{mx} \ -N(0,\sigma_{mx}{}^2)$$
$$v_{my} \ -N(0,\sigma_{my}{}^2)$$
$$v_g \ -N(0,\sigma_g{}^2)$$

With that we recap the following state dynamic equations. Note that the subscripts of t and t+1 denote that the next state is in part determined from the previous state on a discrete time system

$$f_1 = x_{t+1} = x_t + \frac{R_w(\omega_l+\omega_r)}{2} cos(\theta)T$$
$$f_2 = y_{t+1} = y_t + \frac{R_w(\omega_l+\omega_r)}{2} sin(\theta)T$$
$$f_3 = \theta_{t+1} = \theta_t + \frac{R_w(\omega_r-\omega_l)}{2D} T$$
$$f_4 = \theta'_{t+1} = \frac{R_w(\omega_r-\omega_l)}{2D}$$

And the following measurement equations

$$h_1 = d = minimum\ positive(d_1, d_2, d_3, d_4)$$
$$h_2 = l = minimum\ positive(l_1, l_2, l_3, l_4)$$
$$h_3 = m_x = Msin(\theta)$$
$$h_4 = m_y = Mcos(\theta)$$
$$h_5 = g = \theta'$$

The d's and l's are computed in the following way

$$d_1 = \frac{L-x}{cos(\theta)} \qquad l_1 = \frac{L-x}{sin(\theta)}$$
$$d_2 = \frac{W-y}{sin(\theta)} \qquad l_2 = \frac{W-y}{-cos(\theta)}$$
$$d_3 = \frac{x}{-cos(\theta)} \qquad l_3 = \frac{x}{-sin(\theta)}$$
$$d_4 = \frac{y}{-sin(\theta)} \qquad l_1 = \frac{y}{cos(\theta)}$$

Background

The goal of this lab was to provide us a firm understanding of how an Extended Kalman Filter (EFK) works. Kalman Filters (KF) are used to estimate the state of a system based on its inputs and outputs, since states are internal to the system and typically not known by an external observer. KFs act as state estimators by outputting a best guess and its uncertainty at a point in time. These two outputs evolve as new system inputs (actions) and measurement outputs (observations) are put into the state estimator. When a new action is given to the state estimator, these updates are known as dynamics propagation and the values are in priori stage. The following formulas display this relationship

$$\hat{x}_t^{+} : \textit{Best state estimate vector after observation } (\textit{posterior})$$

$$\hat{x}_t^{-} : \textit{Best state estimate vector after action, before observation } (\textit{priori})$$

$$u : \textit{input vector}$$

$$F : \textit{state – transition model matrix}$$

$$G : \textit{control – input model matrix}$$

$$P_t^{+} : \textit{Covariance matrix of best state estimate } (\textit{posterior})$$

$$P_t^{-} : \textit{Covariance matrix of best state estimate } (\textit{priori})$$

$$Q : \textit{Covariance Matrix of Actuator Noise}$$

$$\hat{x}_{t+1}^{-} = F\hat{x}_t^{+} + Gu$$

$$P_{t+1}^{-} = FP_t^{+}F^T + Q$$

When a new observation is given to the state estimator,, these updates are known as measurement update and the values are in the posterior stage. The following formulas display this relationship.

$$K_t : \textit{Kalman Gain Matrix}$$

$$H : \textit{observation model matrix}$$

$$z : \textit{observation vector}$$

$$R : \textit{ Covariance Matrix of Sensor Noise}$$

$$\hat{x}_t^{+} = \hat{x}_t^{-} + K_t(z - H\hat{x}_t^{-})$$

$$P_t^{+} = (1 - K_t H)P_t^{-}$$

$$K_t = P_t^{-}H^T(HP_t^{-}H^T + R)^{-1}$$

The dimensions and values of these matrices depends on the system being analyzed and will be further detailed for our experiment later. The important thing to note, is that KFs work for linear systems and our system is not linear. That is what an EKF is for. The model matrices become the following time dependent matrices

$$F => F_t \quad \textit{jacobian of state dynamics with respect to } (\textit{w.r.t}) \textit{ states}$$

$$G \Rightarrow G_t \quad \textit{jacobian of state dynamics w.r.t inputs}$$
$$H \Rightarrow H_t \quad \textit{jacobian of measurement equations w.r.t states}$$

By taking the jacobians and evaluating them at our current best state estimate, we linearize the system at each iteration so that we can use the dynamics propagation and measurement update equations. Using this knowledge we were able to design a Kalman filter for our system.

Mathematical Formulation of EKF

We begin by noting the dimensions of all our matrices and vectors

$$\textit{number of states}(n) = 4$$
$$\textit{number of inputs}(m) = 2$$
$$\textit{number of observations}(p) = 3$$

$$F_t : n \times n \quad G_t : n \times m \quad H_t : p \times n \quad Q : n \times n \quad R : p \times p \quad P_t : n \times n \quad K_t : n \times p$$
$$\hat{x}_t^- : n \times 1 \quad \hat{x}_t^+ : n \times 1 \quad u : m \times 1 \quad z : p \times 1$$

Note that the number of observations is 3 although the sensors produce 5 values. The reason for this is that the two magnetometer readings were deemed to be too noisy and were not used for the EKF. This can be done because we can estimate our orientation using only the more accurate gyro measurement. Alternatively, we could have included the magnetometer readings for more information and we do encourage this method in the future, but for the sake of time we chose to omit using those observations for our state estimator.

The first thing to do is to calculate the jacobian of the states dynamics w.r.t states and that takes the following form.

$$F_t = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \theta'} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial \theta'} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} & \frac{\partial f_3}{\partial \theta'} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial \theta} & \frac{\partial f_4}{\partial \theta'} \end{bmatrix}_{x = \hat{x}_t^+, u = u_t, w = w_t}$$

Note that all jacobians are evaluated with states being your best guess at that time, inputs at that time and noise at that time. The formulas for the matrix elements are as follows

$\frac{\partial f(below)}{\partial x(right)}$	∂x	∂y	$\partial \theta$	$\partial \theta'$
∂f_1	1	0	$-\frac{R_w(\omega_l+\omega_r)}{2} sin(\theta) T$	0

∂f_2	0	1	$\frac{R_w(\omega_l+\omega_r)}{2} cos(\theta)T$	0
∂f_3	0	0	1	0
∂f_4	0	0	0	0

The next thing to do is to calculate the jacobian of the state dynamics w.r.t inputs and takes the following form and has the following entries

$$G_t = \begin{bmatrix} \frac{\partial f_1}{\partial \omega_l} & \frac{\partial f_1}{\partial \omega_r} \\ \frac{\partial f_2}{\partial \omega_l} & \frac{\partial f_2}{\partial \omega_r} \\ \frac{\partial f_3}{\partial \omega_l} & \frac{\partial f_3}{\partial \omega_r} \\ \frac{\partial f_4}{\partial \omega_l} & \frac{\partial f_4}{\partial \omega_r} \end{bmatrix}_{x=\hat{x}_t^+, u=u_t, w=w_t}$$

$\frac{\partial f\ (below)}{\partial \omega (right)}$	$\partial \omega_l$	$\partial \omega_r$
∂f_1	$\frac{R_w}{2} cos(\theta)T$	$\frac{R_w}{2} cos(\theta)T$
∂f_2	$\frac{R_w}{2} sin(\theta)T$	$\frac{R_w}{2} sin(\theta)T$
∂f_3	$-\frac{R_w}{2}T$	$\frac{R_w}{2}T$
∂f_4	$-\frac{R_w}{2}$	$\frac{R_w}{2}$

Next we need the jacobian of the sensor model w.r.t states and that takes the following form and takes the following form and entries

$$H_t = \begin{bmatrix} \frac{\partial z_1}{\partial x} & \frac{\partial z_1}{\partial y} & \frac{\partial z_1}{\partial \theta} & \frac{\partial z_1}{\partial \theta'} \\ \frac{\partial z_2}{\partial x} & \frac{\partial z_2}{\partial y} & \frac{\partial z_2}{\partial \theta} & \frac{\partial z_2}{\partial \theta'} \\ \frac{\partial z_5}{\partial x} & \frac{\partial z_5}{\partial y} & \frac{\partial z_5}{\partial \theta} & \frac{\partial z_5}{\partial \theta'} \end{bmatrix}_{x=\hat{x}_t^+, u=u_t, w=w_t}$$

$\frac{\partial z (below)}{\partial x (right)}$	∂x	∂y	$\partial \theta$	$\partial \theta'$
∂z_1	A	B	C	0
∂z_2	D	E	F	0
∂z_5	0	0	0	1

The 6 entries denoted as capital letters are because these entries are piecewise functions that are dependent on which d and l was chosen from the minimum positive choices. The following tables detail what entries are used depending on the values of d and l that were chosen. For example, if d1 and l4 were used, then ABC would be the first row of table 1 and DEF would be fourth row of table 2.

Table 1

Min Pos	A	B	C
d_1	$-\frac{1}{cos(\theta)}$	0	$(L-x)*\frac{sin(\theta)}{cos(\theta)^2}$
d_2	0	$-\frac{1}{sin(\theta)}$	$-(W-y)*\frac{cos(\theta)}{sin(\theta)^2}$
d_3	$-\frac{1}{cos(\theta)}$	0	$-x*\frac{sin(\theta)}{cos(\theta)^2}$
d_4	0	$-\frac{1}{sin(\theta)}$	$y*\frac{cos(\theta)}{sin(\theta)^2}$

Table 1: This table is used to determine the elements of the first row of the linearized H_t and is read in the following fashion. If min pos was d_1 then A, B, and C take the functions of row 1 respectively. This table was necessary due to the piecewise nature of the sensor model with respect to d

Table 2

Min Pos	D	E	F
l_1	$-\frac{1}{sin(\theta)}$	0	$-(L-x)*\frac{cos(\theta)}{sin(\theta)^2}$
l_2	0	$\frac{1}{cos(\theta)}$	$-(W-y)*\frac{sin(\theta)}{cos(\theta)^2}$
l_3	$-\frac{1}{sin(\theta)}$	0	$x*\frac{cos(\theta)}{sin(\theta)^2}$
l_4	0	$\frac{1}{cos(\theta)}$	$y*\frac{sin(\theta)}{cos(\theta)^2}$

Table 2: This table is used to determine the elements of the second row of the linearized H_t and is read in the following fashion. If min pos was l_1 then D, E, and F take the functions of row 1 respectively. This table was necessary due to the piecewise nature of the sensor model with respect to l

The only other matrices left to consider are Q and R. These are the covariance matrices and they take the following form and values.

$$G_t = \begin{bmatrix} \sigma_d^2 & 0 & 0 & 0 \\ 0 & \sigma_d^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & \\ 0 & 0 & 0 & \sigma_{\theta'}^2 \end{bmatrix} \quad R_t = \begin{bmatrix} \sigma_d^2 & 0 & 0 \\ 0 & \sigma_l^2 & 0 \\ 0 & 0 & \sigma_g^2 \end{bmatrix}$$

The reason that the non-diagonal are zero is that the noise is assumed to be additive white gaussian noise (AWGN) which means the noise should be independent of other noises so therefore the covariances of different noises is zero

The values of K_t are calculated during every iteration and values of estimate and covariance are initialized to reflect initial conditions and are then updated during every iteration with the KF equations discussed in the previous section. The initial values of of estimate reflect our knowledge of where we think the car is at the beginning and our initial values of covariance P reflect our certainty in those initial conditions. These two values are user dependent and can be changed to emulate different scenarios.

Description of Implementation In Code

There are two main coding implementations for the project. First, the Paperbot.ino is modified to send the sensors data wirelessly from paperbots to a website, without using the serial ports, for calculating variance purposes. However, since there is only a limited amount of data that can be viewed from the website at a time. This put a constraint into how we can develop an accurate variance for noise models. As a result, we discarded the wireless data transmission outputs and estimated our own variances.

The second implementation of code is the Kalman filtering program. First, the base values of the matrices involved in the EKF were initialized. This included the initial state, initial covariance, actuation noise, and sensor noise matrices. Because of the nonlinearity of the EKF, the state dynamics matrices and sensor model matrix needed to be calculated iteratively. In the primary loop of our implementation of the EKF, it first updates the values of Fk and Gk based on the current state. It then uses these to calculate our priori state and covariance matrix. Then, again based on state, the Hk matrix updates, and Kalman gain, posteriori state, and posteriori covariance are calculated. These are then saved to our output file, and the loop repeats. The result, given our input file and simulator output file, is a state list file, which provides the state estimate of the system at each time increment. The matrices of the EKF were calculated using the matrix math tools within the numpy library.

The overall view of stateEstimator.py is to compute states by using Kalman filter algorithm. The algorithm requires input to the car (PWM signals) to compute prior P and X and output from the car (sensors data) to computer posteri P and X. The algorithm then uses a for loop to iterate all those inputs from simulations and outputs P (Variance) and X (Expected Mean) as each estimated state. plot the variance of each estimated state overtime and the offset between its estimated states and the predicted states from simulation. There are two main input files. First, it reads the pwm signals and sensors output predicted from the simulation to feed into kalman filter computation. The second file outputs states from simulation to compare how far off the the kalman state estimator is to the simulation’s states. The stateEstimator program overall plots

eight graphs - four graphs for each of the variances over time (incremented by 0.1 sec), and another 4 graphs for the offset between Kalman estimated states (X, Y, θ, θ') and simulation's.

Experimental Setup

Our test setup was initially to test the state estimator using a fixed trajectory involving straight movement and a turn. This would be analyzed under variance scenarios where the sensor noise and actuator noise varied in combinations of large and small uncertainty. Separately we would also test how the estimator varied under high certainty for initial conditions and high uncertainty for initial conditions. This way we could test the estimator under various conditions and see how that effects variables such as accuracy and certainty of resulting estimate and time of convergence for P. We chose not to focus on hardware comparisons because at this point our hardware system had deteriorated that simulation models would not align and actuator models could not have been re-evaluated in time. We plan to repair the hardware system for testing in the future.

Furthermore, due to issues encountered during implementation of EKF, we had to limit the extent of our experiments to a simple straight trajectory and only under one set conditions, namely a nice actuator and sensor noise model. Initial conditions were set to the center of the arena oriented parallel to x axis with no angular velocity and with a nice level of certainty. For further details on this refer to the constants in simulator and EFK code. This restriction was again due to a lack of time. Future testing will allow for more variables to be manipulated.

Experimental Data and Results

Before official experimentation was completed, tests revealed significant error in the estimated x-state values. In order to quantify these errors, they were plotted alongside the simulation x-states and the difference between the two. The results are plotted below.

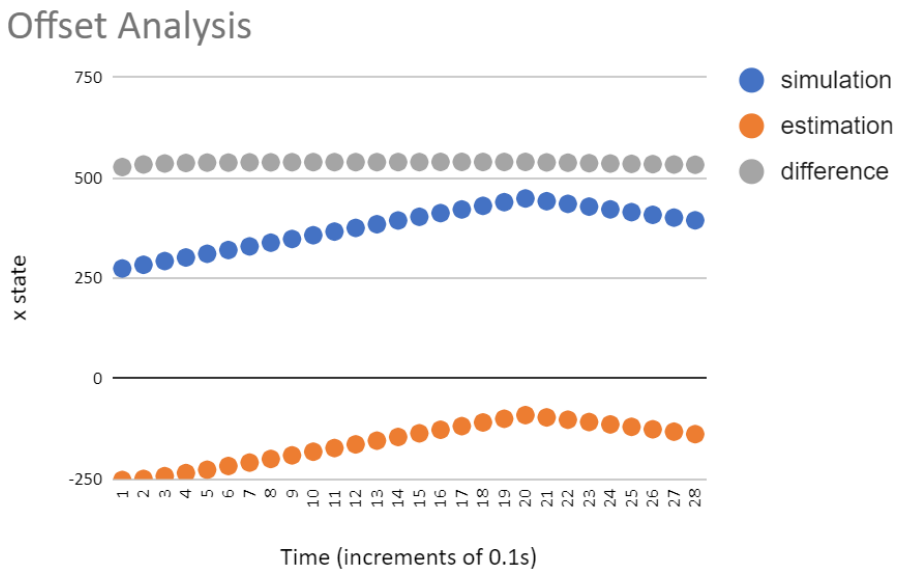

Figure 1: X-state in simulation and estimation and their differences

As is clear in this chart, significant error is present. However, the error is also extremely consistent. As we could not find the source of this error, for the purposes of readability, we have added a constant offset to the estimated x-state to produce more accurate results. In order to test the state estimator, a set of PWM values intended to create a simple, straight-line path was provided to the simulator and the state estimator, the simulator was used to produce sensor output values, and these were also provided to the state estimator. In addition, the state estimator was provided with reasonable estimates of the variance in the initial position, the actuation model, and the sensor model. The resulting x and y states are plotted below.

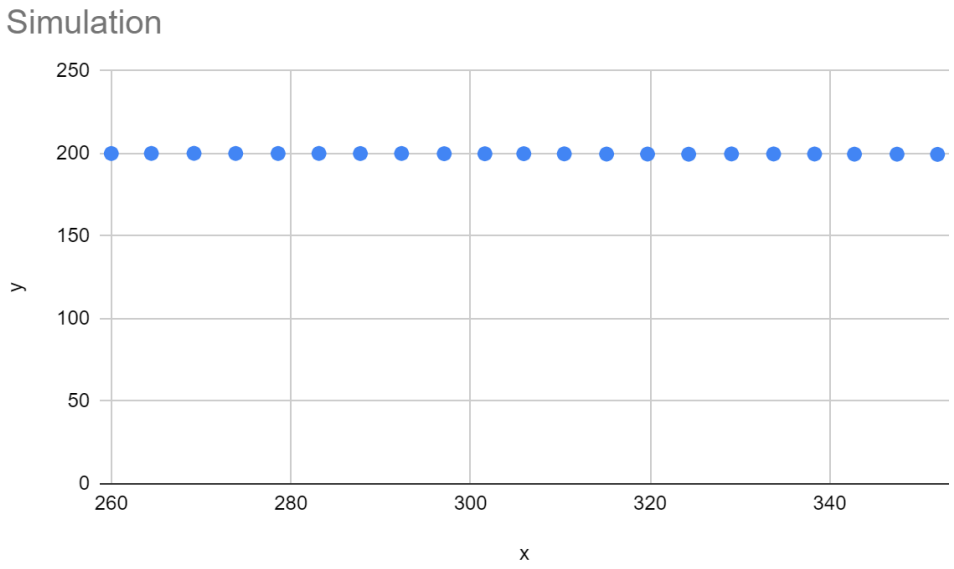

Figure 2: X and Y states from the simulation

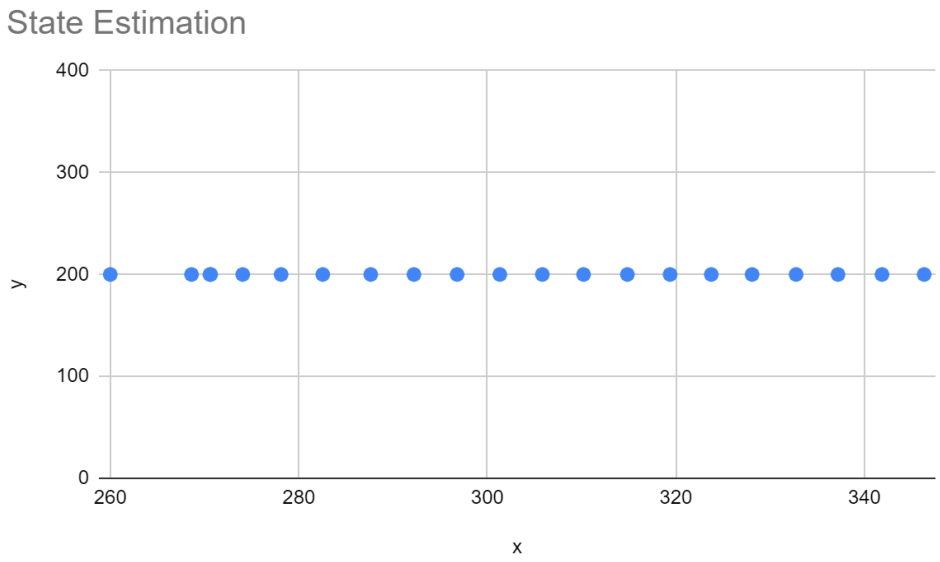

Figure 3: X and Y states from the simulation

In addition, the mean-square error of both theta and theta-dot were calculated to have values of 4.29e-5 and 0, respectively. Given this data, we can confirm a reasonable degree of accuracy in estimating states given this specific situation. Despite other minor issues we ran into during this experiment, we can vouch for the validity and usefulness of the Kalman filter.

Conclusions

In conclusion, although our EKF is still in beta development, the results display that the concept of an estimator exists with the x and y states being estimated well as the two straight line trajectories are maintained and the orientation and angular velocity being kept at a significantly low level. There is still much work to be done in order to get a fully functional EKF implemented in code, but it is nice to see that the bare minimum in expectations can be seen from the results. Much of the trouble in this lab was caused to lab 1 being underdeveloped which spilled into a lot of time for lab 2. Without a hardware system we were forced to rely solely or our original simulator and without proper data-driven models we had to use estimated variance values, but the estimates were reasonable for testing purposes. Hopefully moving forward, we can fully implement the EKF and succeed in lab3.

References:
Github repo-https://github.com/Thequantums/ECE183DA/tree/master/lab2
EE 183-DA lectures:https://www.overleaf.com/7384964663bcvzmtkpvnny