Autonomous Robots:
Assignment 3

Erik Norlin

May, 2023

The Kalman Filter

A Kalman Filter was implemented to estimate the path taken by a differentially steered robot where a ground truth, a GNSS recording as well as estimated wheel speeds were given. Since the case given was non-linear, we used the Extended Kalman Filter (EKF). The algorithm of the EKF is as follows

Predict the next state

$$\hat{x}_k^- = f(\hat{x}_{k-1} + u_k) \tag{1}$$

$$\hat{\mathbf{P}}_k^- = \mathbf{F}_k \hat{\mathbf{P}}_{k-1} \mathbf{F}_k + \mathbf{Q} \tag{2}$$

Correct the predicted state with measurement data

$$\mathbf{K}_k = \hat{\mathbf{P}}_k^- \mathbf{H}_k^T (\mathbf{H}_k \hat{\mathbf{P}}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1} \tag{3}$$

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}(z - h(\hat{x}_k^-)) \tag{4}$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}\mathbf{H}_k)\hat{\mathbf{P}}_k^- \tag{5}$$

Superscript with minus: prediction
Only superscript: estimate

k: time step
x: state vector $[v, \phi, x, y]$ (velocity, yaw angle, position)
u: control signal
f: the process model as non-linear functions
$\mathbf{F}$: jacobian of f
$\mathbf{P}$: covariance error of the state
$\mathbf{Q}$: process noise covariance

z: vector of measurements of the state
h: the observation model as non-linear functions
$\mathbf{H}$: jacobian of h
$\mathbf{K}$: Kalman gain; the weight to give to the measurements
$\mathbf{R}$: measurement noise covariance

The process model is what describes how the state transitions to the next state, for the example, the prediction of kinematics of a differentially steered vehicle, where as the observation model describes how the sensors measures the state. The state variables of the process and observation model do not necessarily have to line up. For example, the process model can predict velocity and position where as the observation model may only measure the position.

The EKF relies on that the error of the covariances are gaussian distributed because this gives an appropriate estimate of the state. For example, if the position of a robot is estimated we would get a gaussian distributed error around the robot, which gives a good idea of where it is; most probably around the center of the error (the mean). If the errors were not gaussian distributed, it would become difficult to make any sense of the estimate. For the errors to be gaussian distributed, the system has to be linear and it is therefore we linearize the non-linearites around the state. This is an approximation that we make.

The process model f of this task is the kinematic equation for a differentially steered robot as following

$$v_k = v_k \tag{6}$$
$$\phi_k = \phi_{k-1} + \dot{\phi}dt \tag{7}$$
$$x_k = x_{k-1} + v\cos(\phi)dt \tag{8}$$
$$y_k = y_{k-1} + v\sin(\phi)dt \tag{9}$$
$$\tag{10}$$

Where v and ϕ are calculated using the given estimated wheel speeds as

$$v = \frac{v_R + v_L}{2} \tag{11}$$
$$\dot{\phi} = \frac{(v_R - v_L)}{2R}dt \tag{12}$$

The linearized system $\mathbf{F}$ of the model f becomes

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \cos(\phi)dt & -v\sin(\phi)dt & 1 & 0 \\ \sin(\phi)dt & v\cos(\phi)dt & 0 & 1 \end{bmatrix} \tag{13}$$

Since we predict the state we also want to predict the covariances of the errors $\mathbf{P}$ between the state variables. We do this by taking the expected value of the covariance between the true state μ and the estimated state.

$$COV(x, \mu) = E((x - \mu)(x - \mu)^T) \tag{14}$$
$$\hat{\mathbf{P}}_k^- = E((\hat{x}_k^- - \mu_k)(\hat{x}_k^- - \mu_k)^T) \tag{15}$$

In a linear case we have

$$\hat{x}_k^- = \mathbf{F}\hat{x}_{k-1} + \mathbf{G}u_k \tag{16}$$

Thus

$$\hat{\mathbf{P}}_k^- = E((\mathbf{F}\hat{x}_{k-1} + \mathbf{G}u_k - \mathbf{F}\mu_{k-1} - \mathbf{G}u_k)(\mathbf{F}\hat{x}_{k-1} + \mathbf{G}u_k - \mathbf{F}\mu_{k-1} - \mathbf{G}u_k)^T) \tag{17}$$
$$= E(\mathbf{F}(\hat{x}_{k-1} - \mu_{k-1})(\mathbf{F}(\hat{x}_{k-1} - \mu_{k-1}))^T) \tag{18}$$
$$= E(\mathbf{F}(\hat{x}_{k-1} - \mu_{k-1})(\hat{x}_{k-1} - \mu_{k-1})^T\mathbf{F}^T) \tag{19}$$
$$= \mathbf{F}E((\hat{x}_{k-1} - \mu_{k-1})(\hat{x}_{k-1} - \mu_{k-1})^T)\mathbf{F}^T \tag{20}$$
$$= \mathbf{F}\hat{\mathbf{P}}_{k-1}\mathbf{F}^T \tag{21}$$
$$\tag{22}$$

Similarly we get the same formula for $\hat{\mathbf{P}}_k^-$ for a non-linear case too. Adding the process noise we get

$$\hat{\mathbf{P}}_k^- = \mathbf{F}\hat{\mathbf{P}}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{23}$$
$$\tag{24}$$

$\hat{\mathbf{P}}_k$ in this task was initialized with all zeros because it is reasonable to assume in this case that we know the initial position of the robot with full certainty, $(x, y) = (0, 0)$ in this case. Choosing $\mathbf{Q}$ is not straight forward because the process model takes an input without considering the physical surroundings, so the process noise can at times be lower and at other times higher. One approach is to assume that the process noise exists and set it to the identity matrix and then fine tune it. One has to be careful though because if the process noise is too small the estimate tends to lag, where as a too large process noise can cause the estimate to follow the measurement too closely. In this assignment a ground truth and an estimate of the wheel speeds over the entire run were given from the start which was exploited by calculating the variances and covariances between the prediction and the ground truth of the shared state variables. The state variables x, y were assumed to depend on each other, and the unknown variance of v was arbitrarily set to 1. The process noise was therefore

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.23 & 0 & 0 \\ 0 & 0 & 0.24 & -0.2 \\ 0 & 0 & -0.2 & 0.29 \end{bmatrix} \tag{25}$$

Similarly for the measurement noise $\mathbf{R}$ the covariances were calculated between the GNSS measurements and the ground truth. The GNSS measured the state variables x, y and were assumed to depend on the each other, variances of the unmeasured state variables were arbitrarily set to 1 here too. The measurement noise became

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.49 & 0.08 \\ 0 & 0 & 0.08 & 0.67 \end{bmatrix} \tag{26}$$

The Kalman gain $\mathbf{K}$ controls how much the algorithm should trust the prediction or the measurements. If $\mathbf{R}$ is zero everwhere, i.e. there is no error from any of the measurements, we get

$$\begin{aligned} \mathbf{K}_k &= \hat{\mathbf{P}}_k^- \mathbf{H}_k^T (\mathbf{H}_k \hat{\mathbf{P}}_k^- \mathbf{H}_k^T)^{-1} && (27) \\ &= \mathbf{H}_k^{-1} && (28) \end{aligned}$$

Correcting the prediction we have

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + \mathbf{H}_k^{-1}(z - \mathbf{H}_k \hat{x}_k^-) && (29) \\ &= \hat{x}_k^- + \mathbf{H}_k z - \hat{x}_k^- && (30) \\ &= z && (31) \\ & && (32) \end{aligned}$$

and

$$\begin{aligned} \hat{\mathbf{P}}_k &= (\mathbf{I} - \mathbf{H}_k^{-1}\mathbf{H}_k)\hat{\mathbf{P}}_k^- && (33) \\ &= 0 && (34) \end{aligned}$$

We in other words trust the measurements completely and are absolutely sure about the state, which makes sense because $\mathbf{R}$ gives no uncertainty. On the other hand if $\mathbf{R}$ is large, $\mathbf{K}$ goes to zero and we end up with $\hat{x}_k = \hat{x}_k^-$ and $\hat{\mathbf{P}}_k = \hat{\mathbf{P}}_k^-$, and trust the prediction completely.

Results

Implementing the EKF yielded an estimate that looks to be closer to the ground truth than the GNSS recording, as can be seen in figure 1. One can therefore say that the EKF must have been implemented sufficiently since the estimate shows a more accurate representation of the ground truth than the GNSS recording by itself. If the the estimate would have been equal to or worse than the GNSS recording then the EKF would lose its functionality, because then one could just as well use the GNSS recording as an estimate.

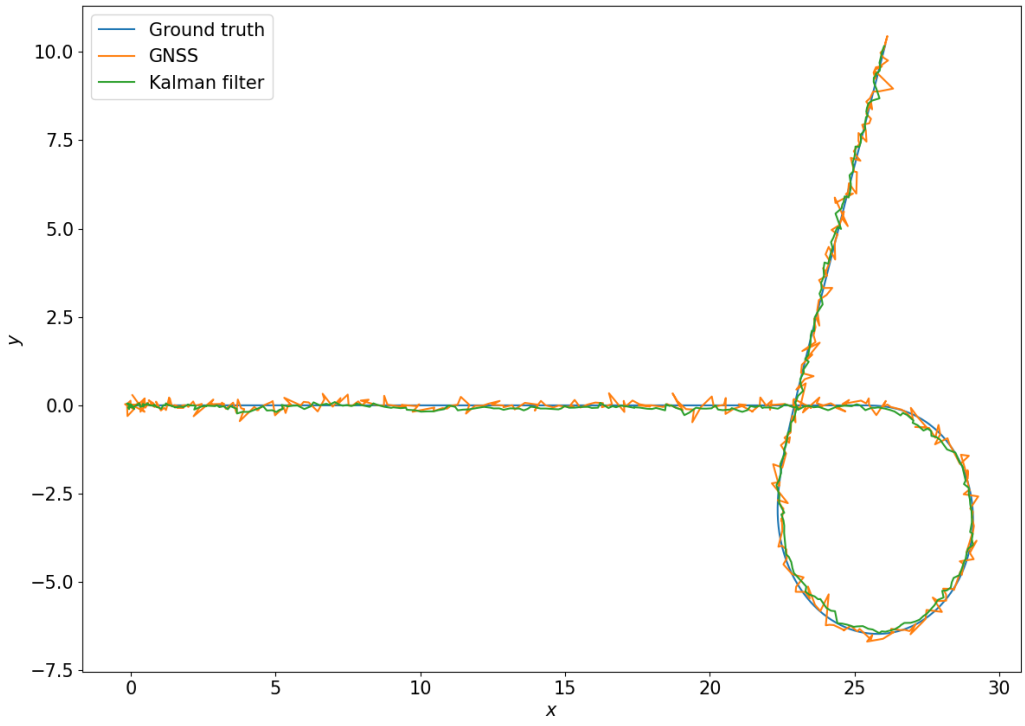

Figure 1: Path taken by the differentially steered robot over 40 seconds. Kalman filter, at sight, shows that it performs better than the GNSS recording.

Looking at the mean square error (MSE) between the estimate and the ground truth in figure 2, it can be noted that the MSE grows over time, especially for x and y but not much for ϕ.

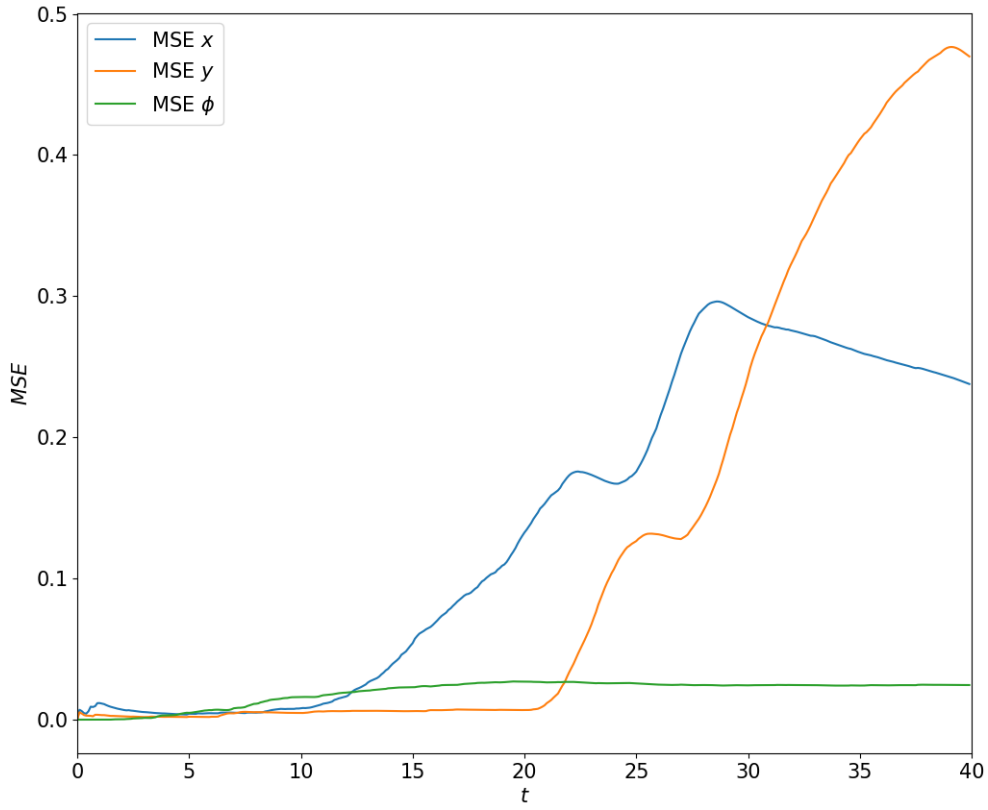

Figure 2: MSE between the estimate and the ground truth. MSE for x and y grows while the MSE for ϕ stays relatively low.

Zooming in on the path taken (figure 3) where the MSE starts to increment and increases until the end (around $15 < t < 40$), it becomes apparent that the EKF has a harder time following along the curvature of the trajectory than the straight line in the beginning of the path. This could have to do with that the process model does not predict the trajectory of the curve accurately, which causes the estimate to deviate further from the ground truth, and therefore increases the MSE. Another reason could be that the noise covariances $\mathbf{Q}$ and $\mathbf{R}$ are not initialized correctly because experimenting with different initializations of these matrices gave different results, sometimes better and sometimes worse. This indicates that it is possible to get closer to the ground truth even though the noise covariances were calculated as accurately as possible. It could be worth investigating more about how to properly initialize these covariances since it is generally known that this is not completely trivial.

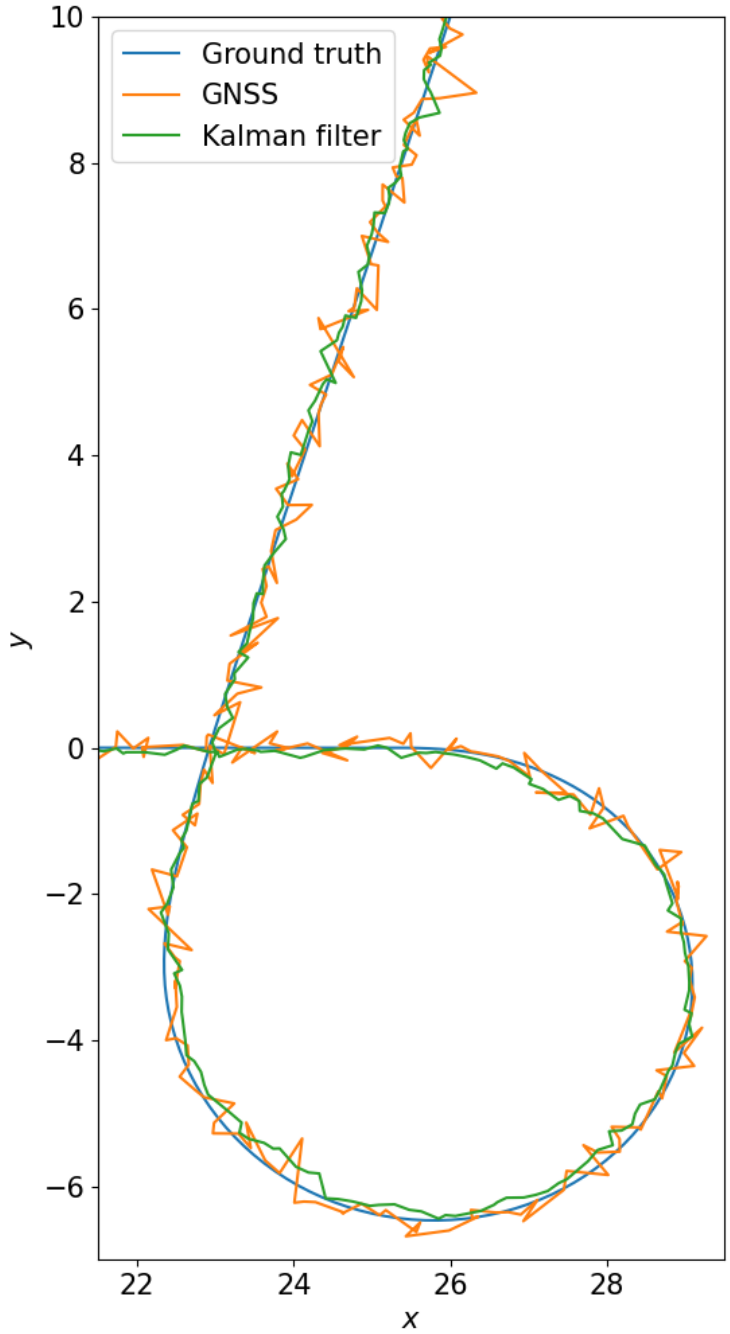

Figure 3: Zoomed in on the path where the MSE starts incrementing until it reaches an all time high ($15 < t < 40$). The EKF estimates the path worse along the curve of the trajectory and then performs better as the path becomes straight again.

An interesting observation when investigating the covariances of $\hat{\mathbf{P}}$ as they change through the run is that the uncertainty of v (figure 4) is by far the largest covariance of all, which indicates

that the process model is quite off when predicting the velocity in comparison to the other state variables. This is not surprising since we do not have any measurements to correct it. Moreover, most covariances are drastically changed between $20 < t < 35$ as can be seen in figure 4, 5, 6 and 7. This could support the argument that the process model performs worse around the curvature of the path. In figure 4 it can be seen that the uncertainty of the velocity has two peaks when following the curvature. This could indicate that the uncertainty of the speed peaks as the robot moves down and then up along the y-axis. In addition to this it can also be noted in figure 7 that the uncertainty between v and y increases drastically within the same interval, which could confirm that uncertainty of the speed along the y-axis increases. A similar observation about x can also be seen in figure 6, though not as drastically as for y.

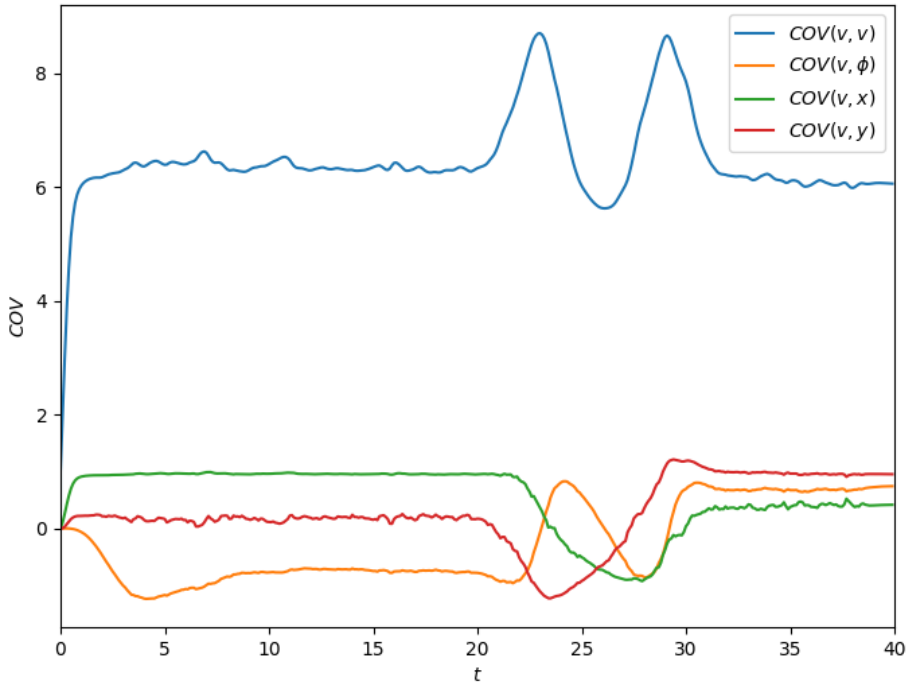

Figure 4

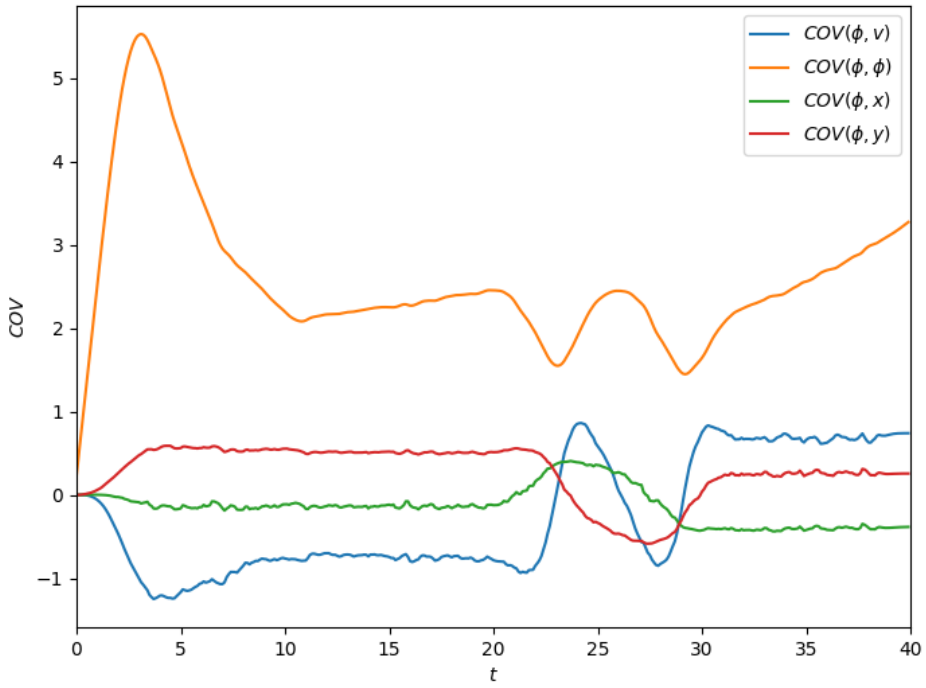

Figure 5

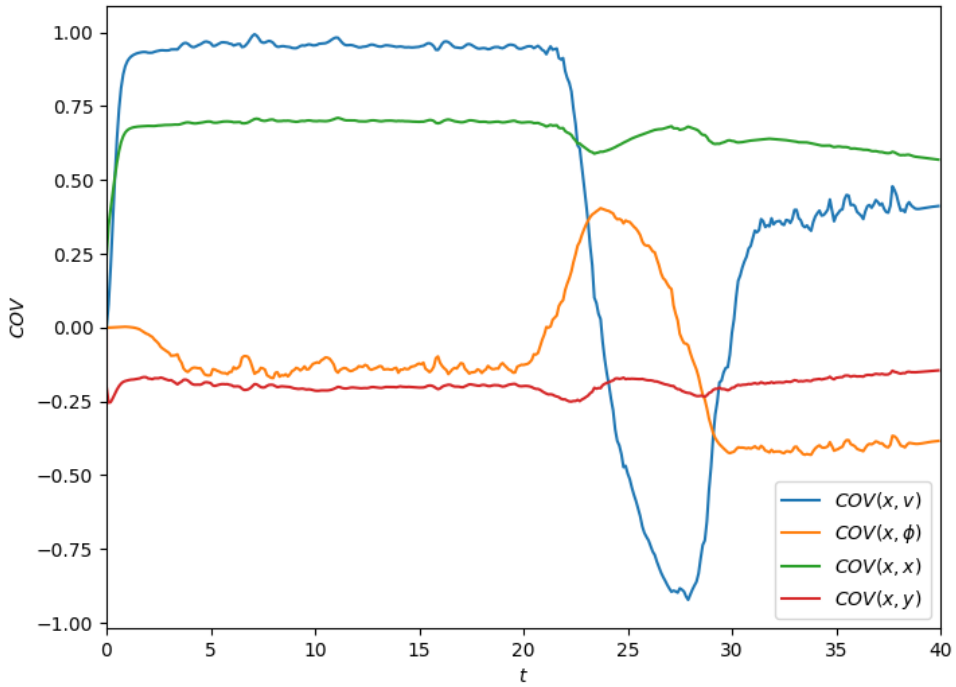

Figure 6

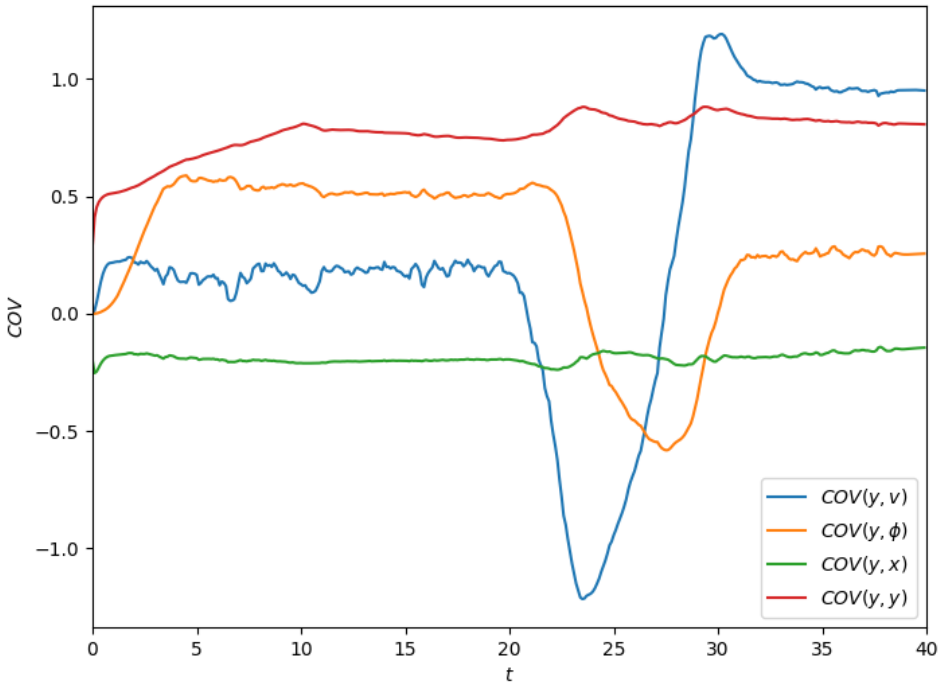

Figure 7

Conclusion

From these observations we can conclude that the process model predicts the position in the direction of y poorly. This can be due to reasons such as wheel slipping or other things that the process model does not recognize. However, the Kalman filter corrects the state with the observation model well enough to be better than the observation model alone. This confirms that we have a Kalman filter that fulfills its purpose and functionality.

Script

To run the script simply run the main script with the command `python3 main.py`. Before running, make sure that all modules load and save data from and to appropriate paths. To plot, use the jupyter notebook.