# Assignment 3 - Kalman filter

TME290 Autonomous robots

Axel Johansson
axejoh@student.chalmers.se

May 18, 2023

---

*Solution proposal*

---

## 1 Kalman filter

The Kalman filter is an efficient recursive filter used to estimate the internal state of a linear dynamic system from a series of noisy measurements. It is widely applied in engineering and econometric fields, including radar, computer vision, and macroeconomic model estimation.

This estimation algorithm combines a process model and an observation model to obtain accurate estimates of a dynamic system's state. It proves particularly useful when dealing with noisy or incomplete measurements. The primary objective of the Kalman filter is to minimize the error between the estimated state and the true state of the system. The filter builds on two main steps: the prediction step and the update step.

### 1.1 Prediction step

In the prediction step, the Kalman filter uses the process model (in this case the kinematic state) of the system to predict the current state based on the previous state estimate as,

$$\hat{x_k} = f(\hat{x}_{k-1}, u_k). \tag{1}$$

It also predicts the covariance of the estimation error. The dynamic model describes how the system evolves over time, taking into account any known inputs or control signals like the wheel speed,

$$P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1}. \tag{2}$$

Where F is the jacobian of f which is used to estimate the non-linear functions. Q is the process noise covariance matrix. It represents the covariance of the process noise, which accounts for the uncertainties and variations in the system dynamics. The error covariance matrix, P, represents the uncertainty or error in the estimate of the system's state. It provides information about the spread and correlations between the different state variables in the estimated state. This is learned by the system and can be initialized to a matrix of choice.

### 1.2 Update Step

In the update step of the Kalman filter, the algorithm incorporates the measurement data to refine the state estimate obtained from the prediction step. This step is essential for improving the accuracy of the estimated state by incorporating the available measurement information.

When the GNSS data is updated then this step is carried out. It begins by comparing the predicted measurement with the actual measurement. The difference between these two measurements is called the measurement residual. The measurement residual represents the discrepancy between the predicted state and the actual measured values.

To incorporate the measurement data, the Kalman filter calculates the Kalman gain

$$G = PH^T(HPH^T + R)^{-1}. \tag{3}$$

The Kalman gain determines the weight given to the predicted state estimate and the measurement in order to obtain an optimal estimate. When the gain is high the systems measurements are accurate but estimates unstable. As can be seen in equation (3) the gain is calculated using the predicted error covariance matrix P, the observation matrix H, and the measurement noise covariance matrix R. H maps the state space (true state) to the measurement space, allowing the Kalman filter to compare the predicted measurement with the actual measurement. R represents the uncertainty or noise associated with the measurement data.

Using the Kalman gain, the update step adjusts the state estimate by combining the predicted state estimate with the measurement. This is done by multiplying the Kalman gain with the difference between the actual measurement and the predicted measurement

$$\hat{x}_k \leftarrow \hat{x}_{k-1} + G_k(z - h(\hat{x}_k)). \tag{4}$$

The Kalman determines the weight assigned to the measurement residual in updating the state estimate. When the Kalman gain is high, it indicates a higher confidence in the accuracy of the measurement data. In such cases, a significant weight is placed on the measurement residual, resulting in a more substantial adjustment to the state estimate.

Furthermore, the update step updates P based on the Kalman gain and H. The updated error covariance matrix represents the refined estimate of the uncertainty in the state estimate after incorporating the measurement data

$$P_k \leftarrow (I - G_k H_k)P_k. \tag{5}$$

### 1.3 Implementation

he Python implementation utilizes various components such as the state vector, measurement vector, control input, dynamic model, and measurement model. The state vector contains the variables representing velocity, angle, and coordinates (x, y). The measurement model corresponds to the GNSS data provided.

The dynamic model defines how the state evolves over time, while the measurement model establishes the relationship between the state and the measurements. To facilitate quick lookup of GNSS data, the timestep of GNSS is converted into a key with corresponding x and y values. The update step is executed when the times in the wheel speed csv align with the GNSS time. At this point, the state estimate is adjusted based on the measurement information.

The implementation of the Kalman filter includes different Q (process noise covariance matrix) and R (measurement noise covariance matrix) values to test if better performance can be achieved by adapting these parameters to the robot's movement. The assumption behind this approach is that sensor accuracy may vary depending on the degree of rotation, allowing for adjustment of the emphasis placed on each sensor. However, it is found that the inaccuracies in measurements appear to be relatively independent of the type of movement since this did not enhance the Mean Squared Error (MSE) significantly.

# 2 Performance

It was a bit unclear in how you wanted the plots but this is how I interpreted it.

## 2.1 Path Visualization - kalman

Displaying a plot showcasing the ground truth, GNSS detections, and the estimated path over the full run.

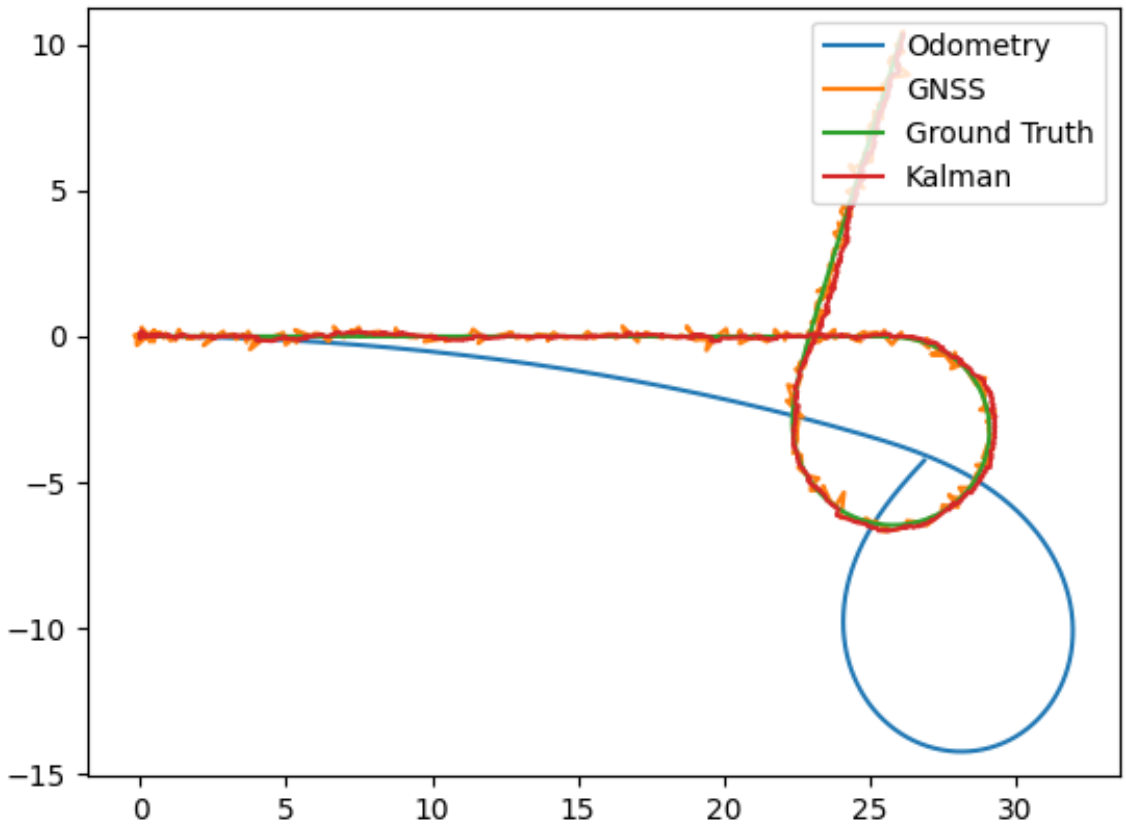


Figure 1: Kalman filter with kinematic state and GNSS.

## 2.2 Mean Square Error

In order to evaluate the accuracy of the estimated values in comparison to the ground truth values, a mean square error (MSE) metric was employed. The computation of MSE values is done as the increments progressed over the full run. Where a loop was executed for each index i in the range of the the dataset, allowing the calculation of the MSE at each step. The MSE summed up to 74,669 and a plot over time can be seen in figure 3.

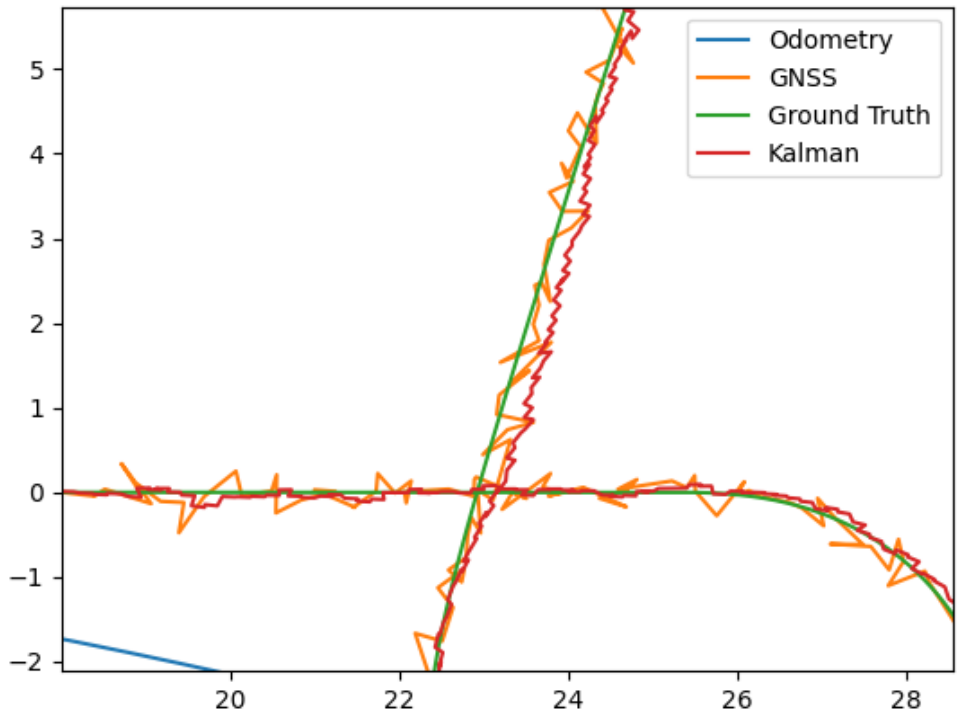


Figure 2: Largest error is after the turn.

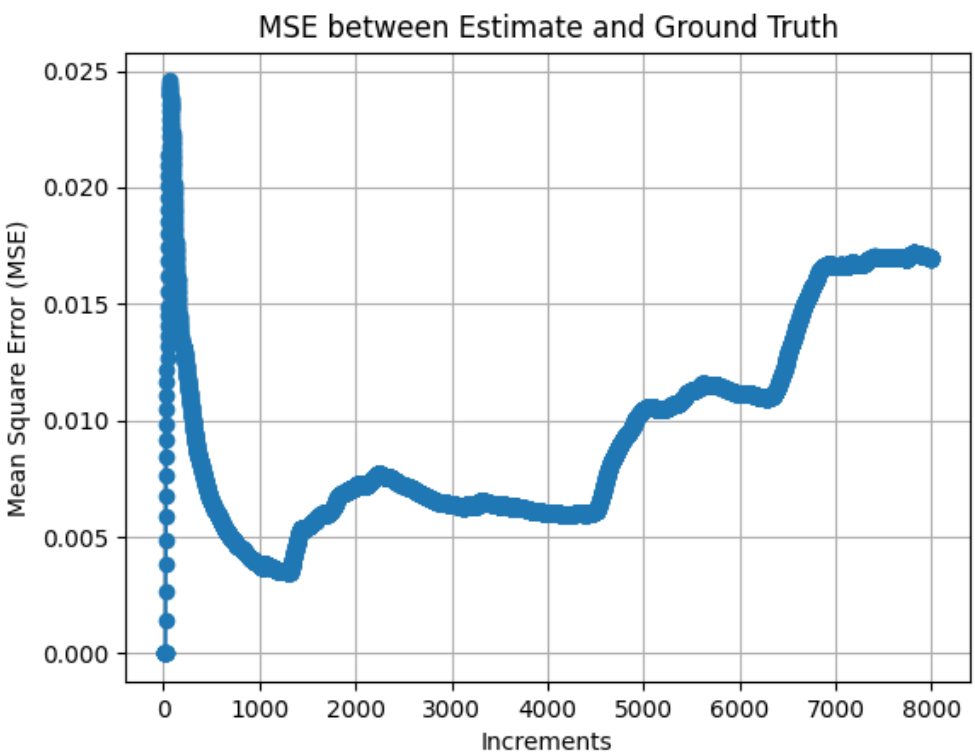


Figure 3: MSE. Its like you can see the 4 timesteps where a behaviour is changed.

### 2.3 Covariance Values

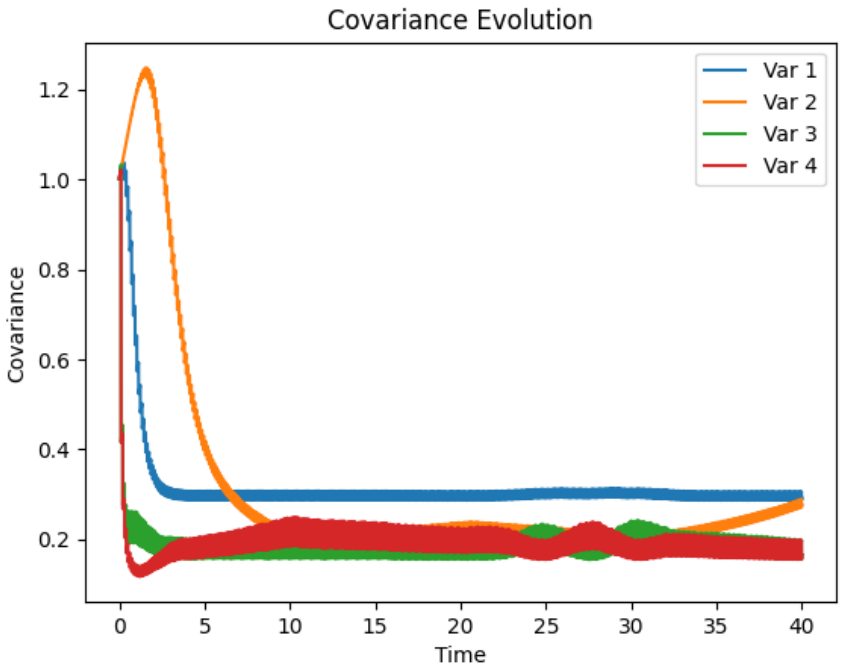


Figure 4: Caption

*Note: I did not do this in c++ because i prioritized the understanding of the Kalman filter.*