# MTE 544 Lab 3 Report

| Stephen Wang | Ajit Rakhra | Howard Dong |
|---|---|---|
| x2274wan | a3rakhra | h53dong |
| 20821034 | 20832703 | 20847639 |

Station 3, Robot 4 on Fri Nov 17 at 8:30 AM

## Understanding the Extended Kalman Filter

The principle of the EKF is that if the motion model describing how the previous robot state evolves to the next is nonlinear, then the covariance matrix of the state variables, modeled as random Gaussian variables, can be calculated using the Jacobian of the motion model, which linearizes the model about the mean of the estimated state. The same goes for the measurement model. This Bayesian filter is aptly named extended Kalman filter because if the motion model and measurement model were linear, the EKF is identical to the original linear Kalman filter.

Here is the motion model of the TurtleBots.

$$\bar{\xi}_{k+1} = f(\xi_k, u)$$

$$\begin{bmatrix} \bar{x}_{k+1} \\ \bar{y}_{k+1} \\ \bar{\theta}_{k+1} \\ \bar{\omega}_{k+1} \\ \bar{v}_{k+1} \\ \bar{\dot{v}}_{k+1} \end{bmatrix} = \begin{bmatrix} f_1(\xi_{k+1}) \\ f_2(\xi_{k+1}) \\ f_3(\xi_{k+1}) \\ f_4(\xi_{k+1}) \\ f_5(\xi_{k+1}) \\ f_6(\xi_{k+1}) \end{bmatrix} = \begin{bmatrix} x_k + v_k \cdot \cos\theta_k \cdot \Delta t \\ y_k + v_k \cdot \sin\theta_k \cdot \Delta t \\ \theta_k + \omega_k \cdot \Delta t \\ \omega_k \\ v_k + \dot{v}_k \cdot \Delta t \\ \dot{v}_k \end{bmatrix}$$

Where $v$ is the speed in the robot heading, in another words the X axis in the robot frame. The effect of control input $u$ is ignored in the model and it's assumed that within a $\Delta t$, the change in the robot state is very small so only the inertial effects are accounted for.

The Jacobian matrix can then be calculated as

$$J_{ij}^{(k)} = \frac{\partial f_i}{\partial \xi^{(j)}} \Big|_{\xi=\xi_k}$$

Where $J_{ij}^{(k)}$ is the element in row $i$ and column $j$ of the Jacobian matrix at time step $k$. Only one examples in the derivation is shown below for brevity.

$$\frac{\partial f_1}{\partial \xi_k^{(3)}} = \frac{\partial}{\partial \theta_k}(x_k + v_k \cdot \cos\theta_k \cdot \Delta t) = -\sin\theta_k \cdot v_k \cdot \Delta t$$

Once $\theta_k$ and $v_k$ are substituted with their estimated means, $J_{13}^{(k)} \cdot \theta_k$ can approximate the original $f_1(\theta_k)$ well, near $\theta_k$. This approximation can then be used to determine the covariance, in other words the preciseness, of the random variable $\theta$ at the next iteration to itself, and to other elements of $\xi$.

The measurement model of the TurtleBot is derived as follows.

$$\bar{z}_{k+1} = h(\bar{\xi}_{k+1})$$

$$\begin{bmatrix} v'_{k+1} \\ \omega'_{k+1} \\ a'_{x,k+1} \\ a'_{y,k+1} \end{bmatrix} = \begin{bmatrix} h_1(\bar{\xi}_{k+1}) \\ h_2(\bar{\xi}_{k+1}) \\ h_3(\bar{\xi}_{k+1}) \\ h_4(\bar{\xi}_{k+1}) \end{bmatrix} = \begin{bmatrix} \bar{v}_{k+1} \\ \bar{\omega}_{k+1} \\ \bar{\dot{v}}_{k+1} \\ \bar{v}_{k+1} \cdot \bar{\omega}_{k+1} \end{bmatrix}$$

Where $v'_{k+1}$ is the expected speed in the robot frame X axis, $a'_{x,k+1}$ is the expected acceleration in the expected odom/global frame X axis measurement at step $k+1$, and $a'_{y,k+1}$ is the expected acceleration in the expected odom/global frame Y axis measurement at step $k+1$.

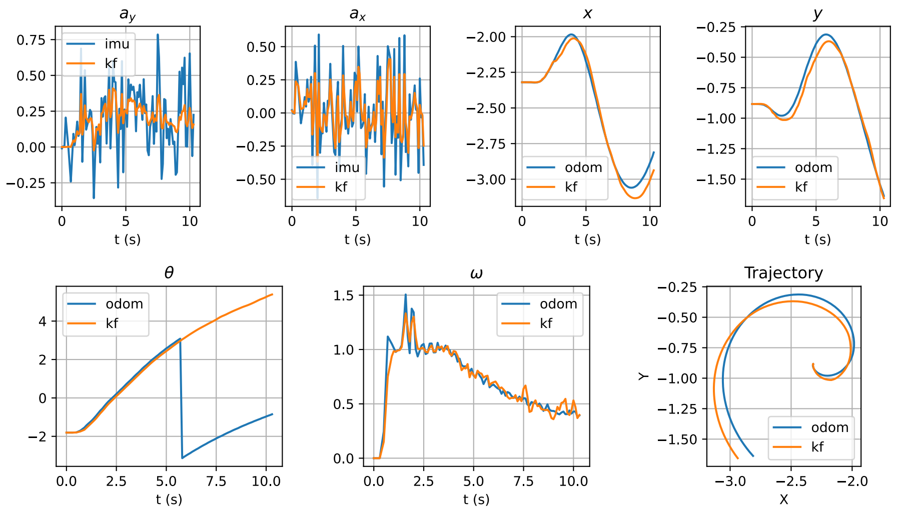
Similar to the Jacobian in the motion model, a partial derivative multiplied with the variable gives a good approximation to the function $h(\bar{\xi}_{k+1})$. One example is shown again for brevity.

$$\frac{\partial h_4}{\partial \bar{\xi}_{k+1}^{(4)}} = \frac{\partial}{\partial \omega_{k+1}}(\bar{v}_{k+1} \cdot \bar{\omega}_{k+1}) = \bar{v}_{k+1}$$

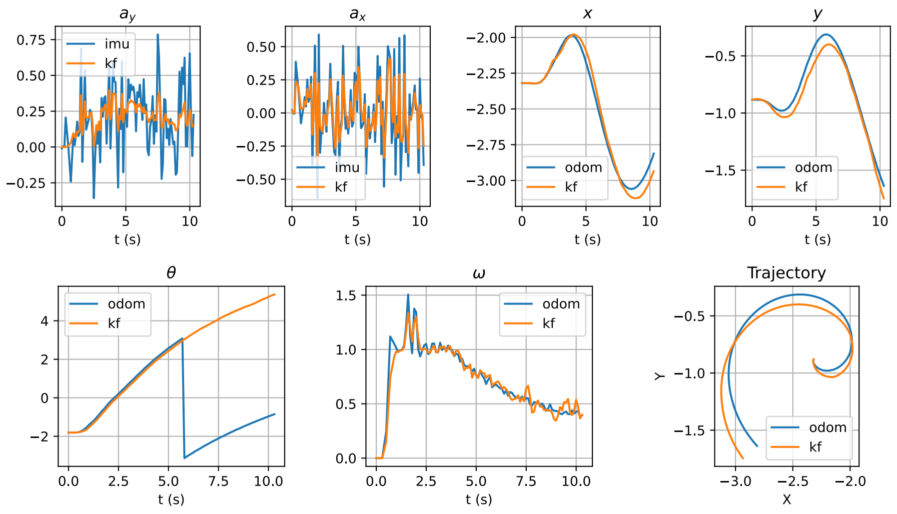
Now the approximation can be used to calculate Kalman gain.

## Evaluation of EKF on TurtleBot3

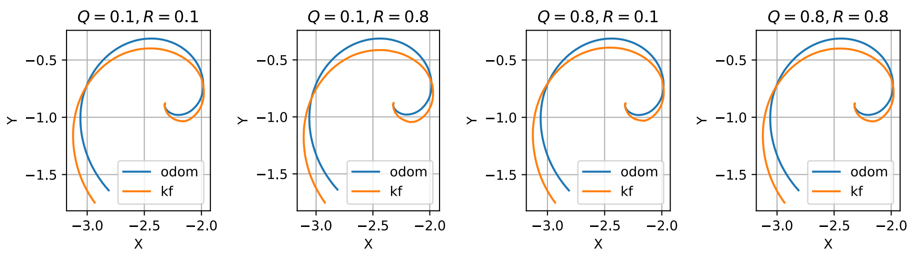
During the lab, we set $Q = 0.5I_6$ and $R = 0.5I_4$. The following figures plot the filter input from IMU measurements and Odom estimates against the filter output.

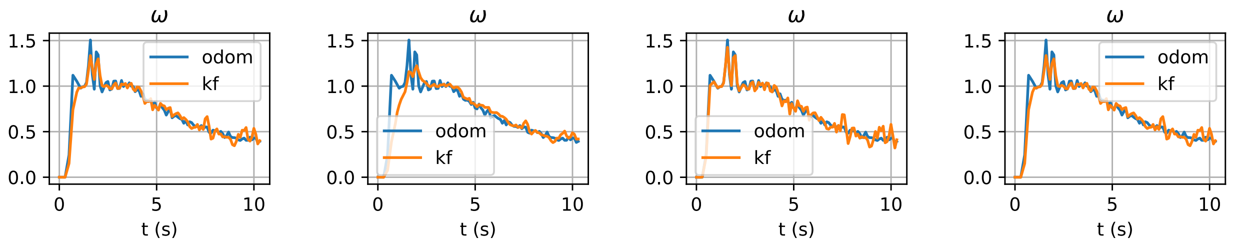


We then varied $Q$ and $R$ to observe the effects of changing the process and measurement covariance matrices on the output. To isolate all variables other than $Q$ and $R$, we used the same set of $z$ obtained from the real robot. The caveat here is that $v$ estimation from Odometry was not recorded so it was reconstructed using a parabola as the mean, referencing to the Kalman filter $v$ output, then Gaussian noise was added on top. Since the objective is to compare the difference in output when $Q$ and $R$ changes, using 3 authentic data and 1 synthetic data should suffice. The following graph shows that the simulated offline result is very similar to the original output.



The following graph shows the comparison in the localization output of the filter for the 4 different combinations where $Q \in \{0.1I_6, 0.8I_6\}$ and $R \in \{0.1I_4, 0.8I_4\}$. The other plots against time is omitted because they are almost indistinguishable.





## Observation From Spiral Motion

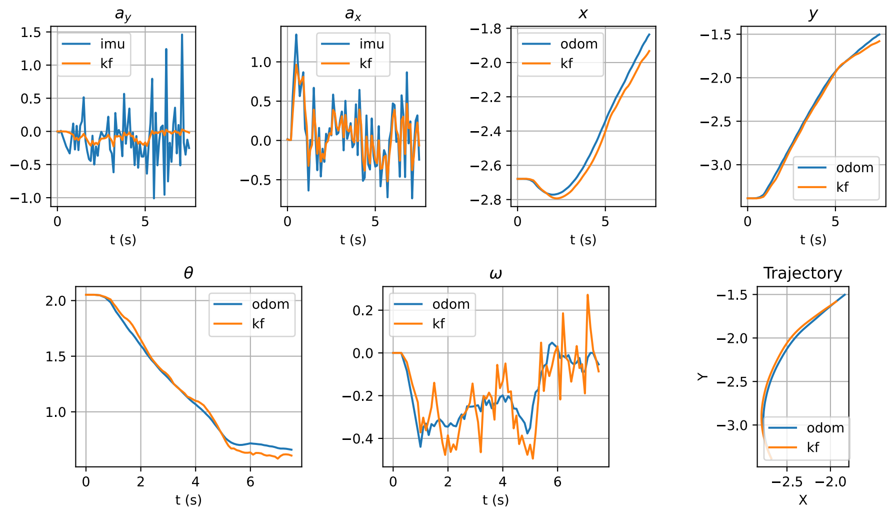
Although the final localization results are mostly indistinguishable to the naked eye, plotting one of the states, $\omega$, over time can provide intuition behind the interaction between the motion model covariance $Q$ and the sensor model $R$.

More succinctly, when $Q \ll R$, the Kalman filter is effectively instructed to treat the sensor measurement to be "less trustworthy"; as a result, the plot for $Q = 0.1, R = 0.8$ shows that the filter output didn't follow the spikes in the measurement closely. There is also a noticeable phase offset, ie lag, in the filter output compared to the Odom input.

Conversely, the plot for $Q = 0.8, R = 0.1$ shows that the filter output followed the sudden fluctuations in the measurement, and sometimes even amplified the noise, since $Q \gg R$.

## Point Navigation

We chose to use $Q = 0.5, R = 0.5$ for the final point navigation test because it seemed to strike a good balance between phase shift (lag) and noise rejection. The following plot shows the filter output against some sensor measurement for reference.



## Conclusion from Point Navigation

The filtered trajectory followed closely with the Odom estimation. Some states like $a_y$ is significantly less noisy compared to the filter input, while some states like $\omega$ became more noisey, surprisingly. More time could be spent characterizing both the model fidelity and the sensor accuracy so that the filter output can be smooth enough to be consumed by a subsequent decision-making algorithm for the robot like obstacle avoidance.