CS8803: STR Spring 2018. Lab 3: Kalman Filtering

Kehinde O. Aina

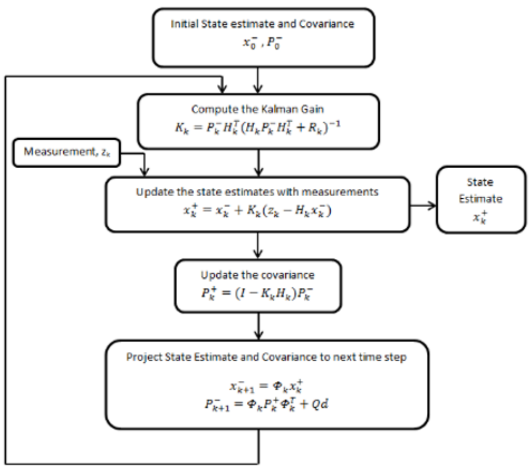
April 11, 2018

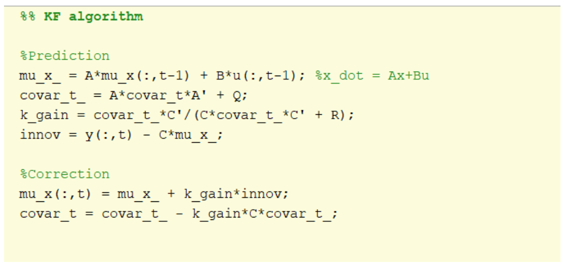
1 Approach

The Kalman Filter produces the optimal estimate for a linear system. It uses a system's dynamics model, known as control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using only one measurement. The approach taken in this lab is to develop a Kalman Filter for the provided linearized helicopter dynamics equation around a stationary point. This involves tuning the process noise and measurement noise parameters to achieve a stable hovering about the stationary point. As we will see in the results below, using an estimate taken directly from the observation makes the filtering to be bias towards measurement noise, while implementing a Kalman Filter for the state estimate seem to equally weigh the effect of both process and measurement noise.

2 Implementation

The recursive algorithm and equations as well as MATLAB code snipped used to implement the Kalman Filter is as shown in Figure 1 below:


Initial State estimate and Covariance
x_0^-, P_0^-
Compute the Kalman Gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$
Measurement, z_k
Update the state estimates with measurements
$x_k^+ = x_k^- + K_k(z_k - H_k x_k^-)$
State Estimate
x_k^+
Update the covariance
$P_k^+ = (I - K_k H_k) P_k^-$
Project State Estimate and Covariance to next time step
$x_{k+1}^- = \Phi_k x_k^+$
$P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + Qd$



```
%% KF algorithm

%Prediction
mu_x_ = A*mu_x(:,t-1) + B*u(:,t-1); %x_dot = Ax+Bu
covar_t_ = A*covar_t*A' + Q;
k_gain = covar_t_*C'/(C*covar_t_*C' + R);
innov = y(:,t) - C*mu_x_;

%Correction
mu_x(:,t) = mu_x_ + k_gain*innov;
covar_t = covar_t_ - k_gain*C*covar_t_;
```

Figure 1: Flowchart and code snippet implementation of Kalman Filter

The first two parts of the lab is completed using an estimate taken from the observation (as provided in the code). The parameters of the algorithm are tuned by modifying the values of sigmaX and sigmaY which are supposed to be the process noise and measurement noise parameters respectively. It is clear that increasing the sigmaY value directly affects the hovering of the helicopter and values above 0.5 makes the helicopter unstable about the stationary point.
The last two parts which involves tuning the process noise and the measurement noise variance matrices, as well as the initial state covariance matrix seem to produce better stationary hovering result, using the Kalman filter state estimator.

2.1 Prediction

First, the current state and the error covariance matrix are predicted at time t. The filter will try to estimate the current mean state conditioned on all the previous mean state: $\hat{\boldsymbol{\mu}}_{t+1} = \boldsymbol{F}\hat{\mu}_t + \boldsymbol{B}u_t$

Next is to estimate the a priori state covariance matrix conditioned on the previous state covariance matrix: $\hat{\boldsymbol{P}}_{t+1} = \boldsymbol{F}\boldsymbol{P}_t\boldsymbol{F}^T + \boldsymbol{Q}_t$ This matrix is used to estimate how much the algorithm should trust the current values of the estimated state. The smaller the value, the more the algorithm trusts the current estimated state.

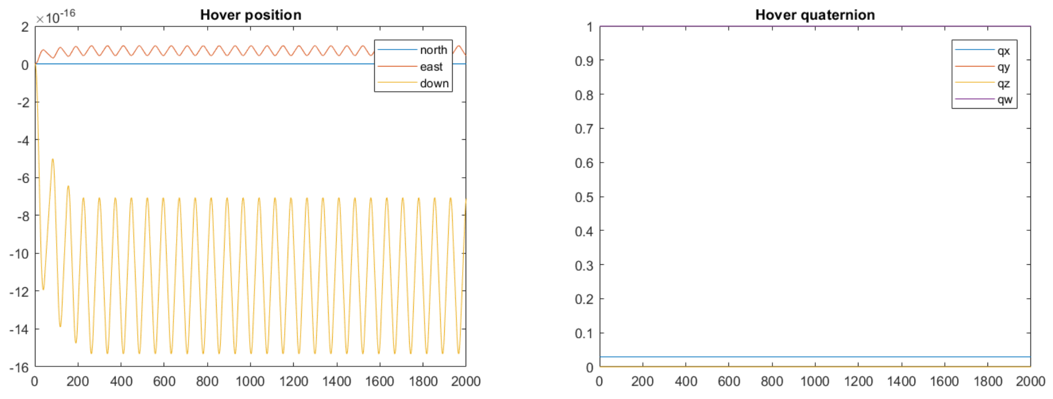
2.2 Update

Here I compute the difference between the measurement y_t and the previously computed mean state $x_{t|t-1}$, which is called the innovation: $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{C}\hat{\mu}_{t+1}$. The observation model C is used to map the a priori state x_t into the observed space. Next is to calculate the innovation covariance: $\boldsymbol{S}_t = \boldsymbol{C}\hat{\boldsymbol{P}}_{t+1}\boldsymbol{C}^T + \boldsymbol{R}$. What it does is that it tries to predict how much the algorithm should trust the measurement based on the error covariance matrix prediction $\hat{P}_{t+1}$ and the measurement covariance matrix R. The observation model C is used to map the a priori error covariance matrix prediction into the observed space. The bigger the value of the measurement noise the larger the value of S, this means that the algorithm should not trust the incoming measurement that much. The Kalman gain is used to to indicate how much we trust the innovation and is defined as: $\boldsymbol{K}_t = \hat{\boldsymbol{P}}_{t+1}\boldsymbol{C}^T\boldsymbol{S}_t^{-1}$ If we do not trust the innovation that much the innovation covariance S will be high and if we trust the estimate of the state then the error covariance matrix P will be small and the Kalman gain will be small and vice versa if we trust the innovation but does not trust the estimation of the current state. The transpose of the observation model C is used to map the state of the error covariance matrix P into observed space, then I compared the error covariance matrix by multiplying with the inverse of the innovation covariance S. Finally, the posteriori estimate of the current state and the posteriori error covariance matrix update are computed as:
$\boldsymbol{\mu}_{t+1} = \hat{\boldsymbol{\mu}}_{t+1} + \boldsymbol{K}_t\,\tilde{\boldsymbol{y}}_t$
$\boldsymbol{P}_{t+1} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C})\hat{\boldsymbol{P}}_{t+1}$.

3 Results

1. Set both error terms (sigmaX, sigmaY in the code) to zero, run the simulation, and observe the successful hovering of the helicopter.

Setting the parameters to zero implies high level of certainty in the sensor readings and thus provides a stable or bounded hovering plot as shown in figure 2 below.



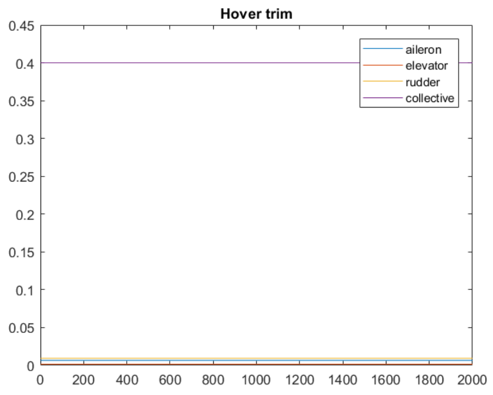
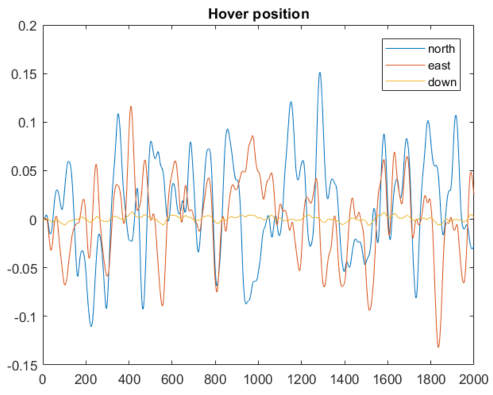


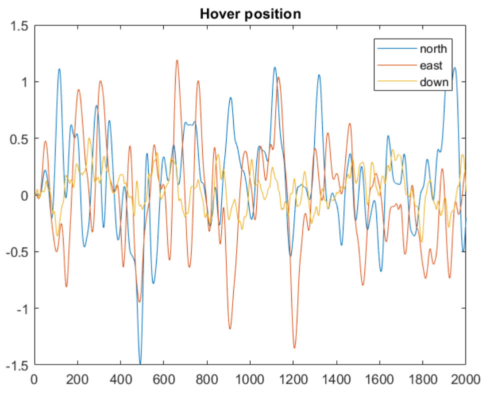
Figure 2: Hovering Helicopter with Zero Noise

2. Increase each error (separately) until it fails to hover. Describe why is it failing and show the NED (North East Down) graph for each case.

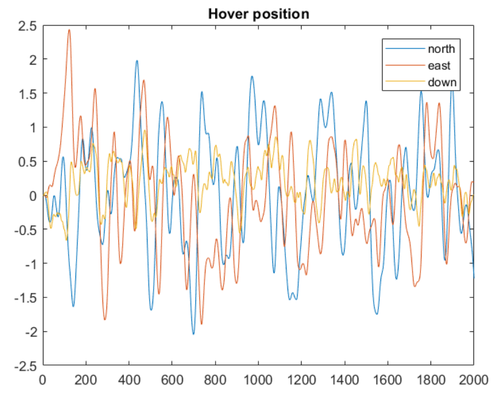
The series of graphs in figure 3 below shows how the hovering varies by changing the parameter. Since the mean state estimate is taken directly from the observation, it follows that the value of sigmaY has greater effect on the hovering state than sigmaX. In fact it turns out that the helicopter starts failing for any value of sigmaY greater than 0.5 (while sigmaX can be set to any value). This is because such a high varince in the measurement suggests a high level of uncertainty or noise in the observation which causes the the system fail by growing exponentially with time.



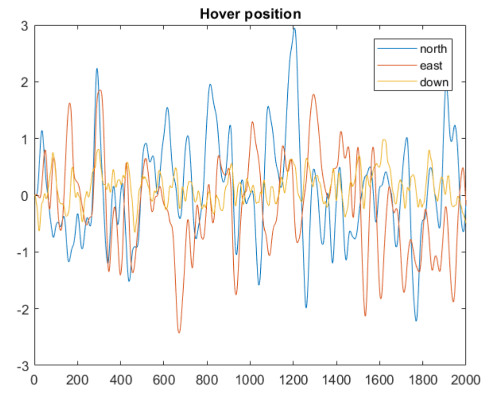
sigmaX = 0.1, sigmaY = 0.1



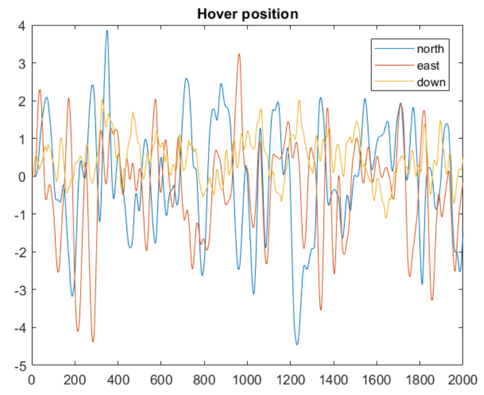
sigmaX = 0.1, sigmaY = 0.2



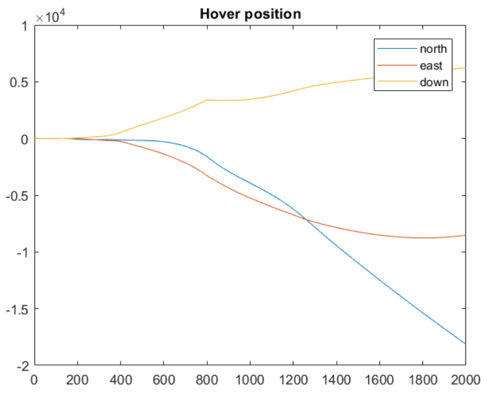
sigmaX = 0.1, sigmaY = 0.3



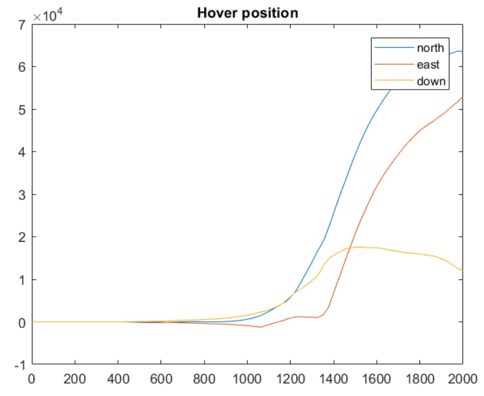
sigmaX = 0.1, sigmaY = 0.4



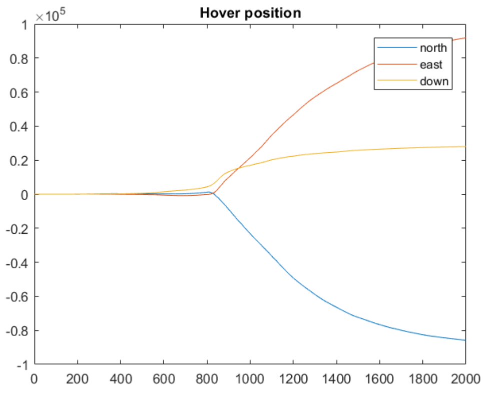
sigmaX = 0.1, sigmaY = 0.5



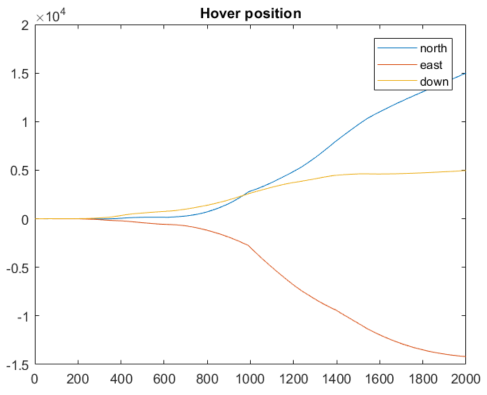
sigmaX = 0.1, sigmaY = 0.6



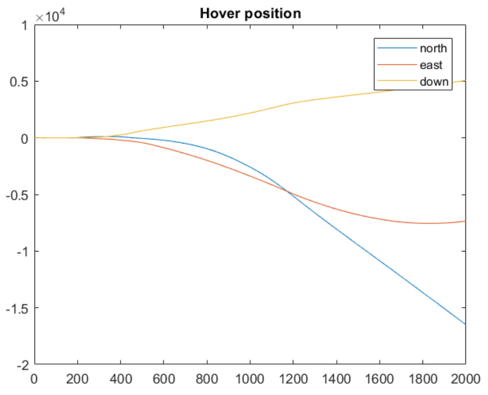
sigmaX = 0.2, sigmaY = 0.6



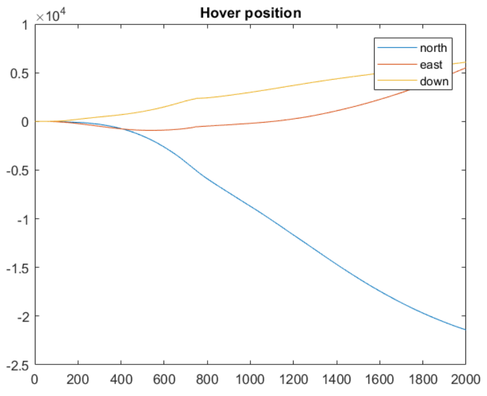
sigmaX = 0.3, sigmaY = 0.6



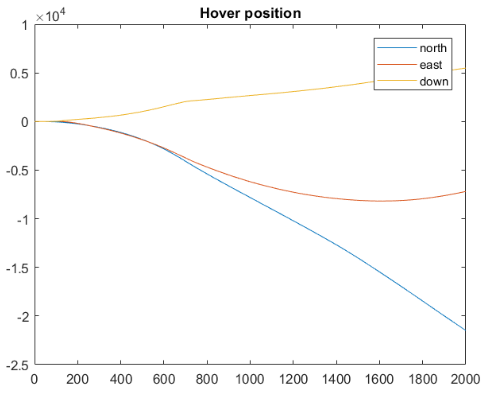
sigmaX = 0.4, sigmaY = 0.6



sigmaX = 0.5, sigmaY = 0.6



sigmaX = 0.6, sigmaY = 0.6



sigmaX = 1, sigmaY = 1

Figure 3: Hovering Helicopter with varying Process Noise and Measurement Noise

3. Reset each error to the default values and implement a Kalman Filter. Tune your parameters so that it can successfully hover. Show all 3 graphs for this.

Figure 4 below show the graphs obtained when the Kalman filter is implemented with default noise parameter values (sigmaX =0.1, sigmaY = 0.5) and the initial covarince matrix is properly selected (0.1 times identity matirx). The parameters are used to generate the positive definite Q and R matrices (identity matrices multilied by sigma-squared) which are used to compute the Kalman gain via the process described in the section above. As the graph shows, the helicopter hovering is bounded within $+/-3$ units of the North, East and Down state variables, and the error is not wholly biased towards the sigmaY value, but both sigmaX and sigmaY contribute to the hovering stability. Similarly, the Hover trim and Hover quaternion are bounded within reasonable limits. Figure 5 however shows a great improvement in the hovering state when the sigmaX and sigmaY values are both set to 0.01 - meaning the white noise is very small. This ensures that the kalman gain and the innovation covariance are equally considered without explicitly preferring one to the other. The hover position is within 0.15 range of the stationary value, while the hover quaternion values are close to zero.

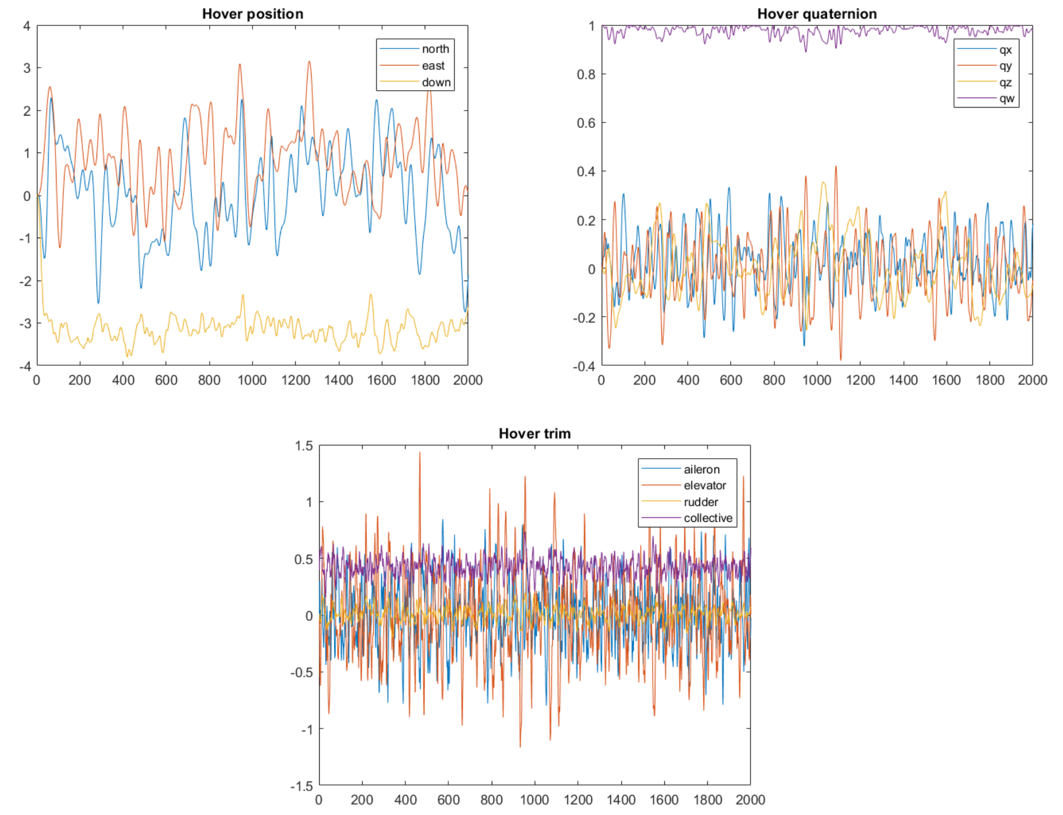
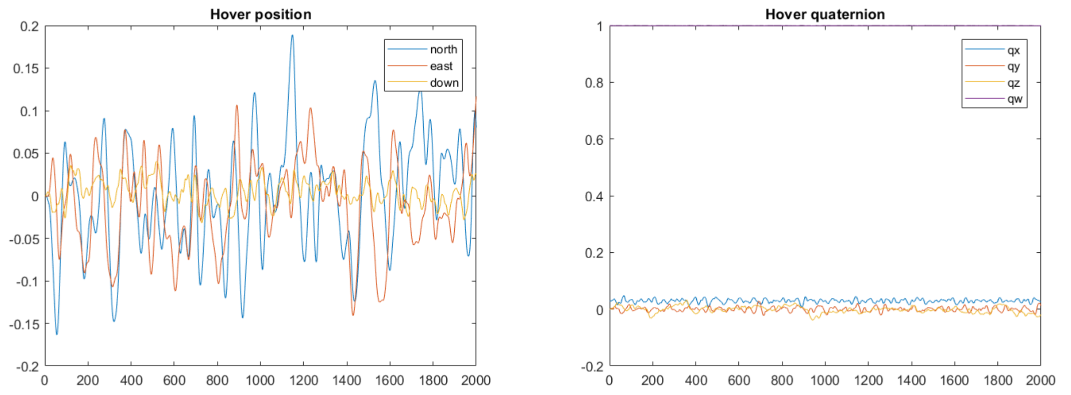


Figure 4: Kalman Filter implementation with default values of Process and Measurement Noise



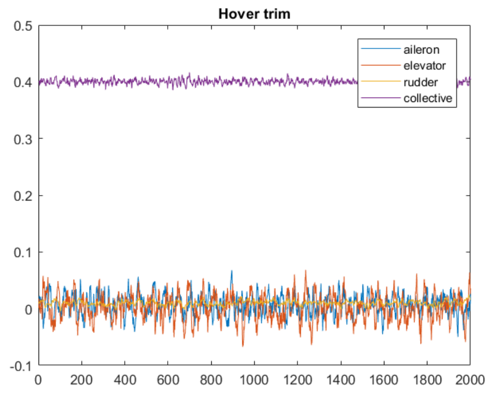


Figure 5: Kalman Filter implementation with tuned values of Process Noise, Measurement Noise and Initial Covariance Matrix

4. Increase each error until it fails and ponder the shortcomings of this system.

The failed hovering position is as shown in figure 6 below. The system failed because of the inaccurate estimate of linearized dynamics and due to the large noise in the measurement. The large noise simulated in the process noise indicates that the linearized model does not represent the true dynamics of the system which makes the kalman filter fail to hover the helicopter. Also the excess error in the system makes the helicopter stray too far from the stationary point.
A solution could be to use the EKF in order to capture the inaccuracies in the estimated dynamics and reduced the process noise, and to get a better sensor with lower measurement noise.

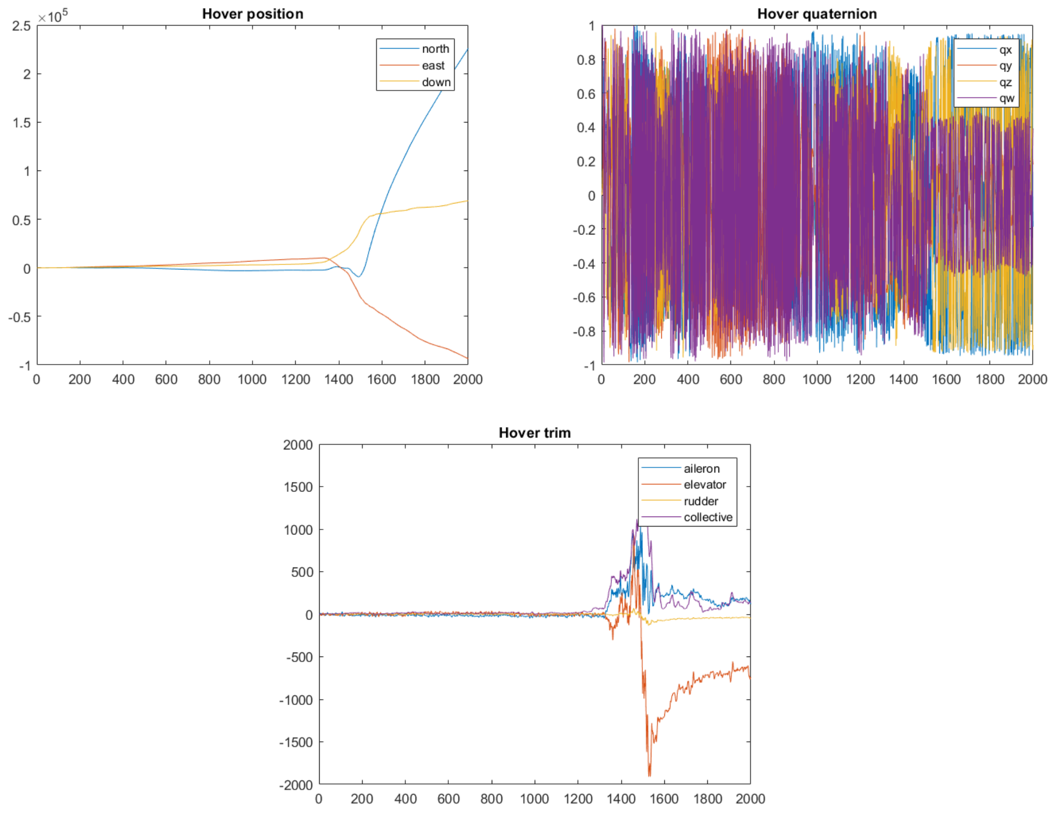


Figure 6: Kalman Filter implementation with increased Process and Measurement Noise

4 Appendix

link to the source code and binary file.