CS8803: STR Spring 2018. Lab 3: Kalman Filter

Thanakorn Khamvilai

April 11, 2018

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

The results after setting both sigmaX and sigmaY to zero are shown in the following figures.

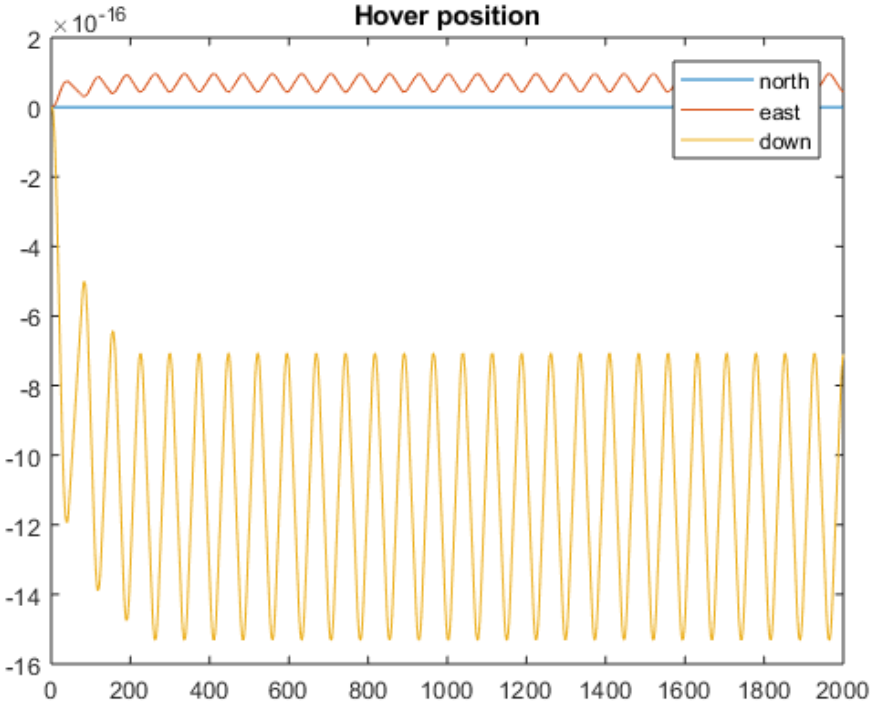

Figure 1: Hover Position without Noise

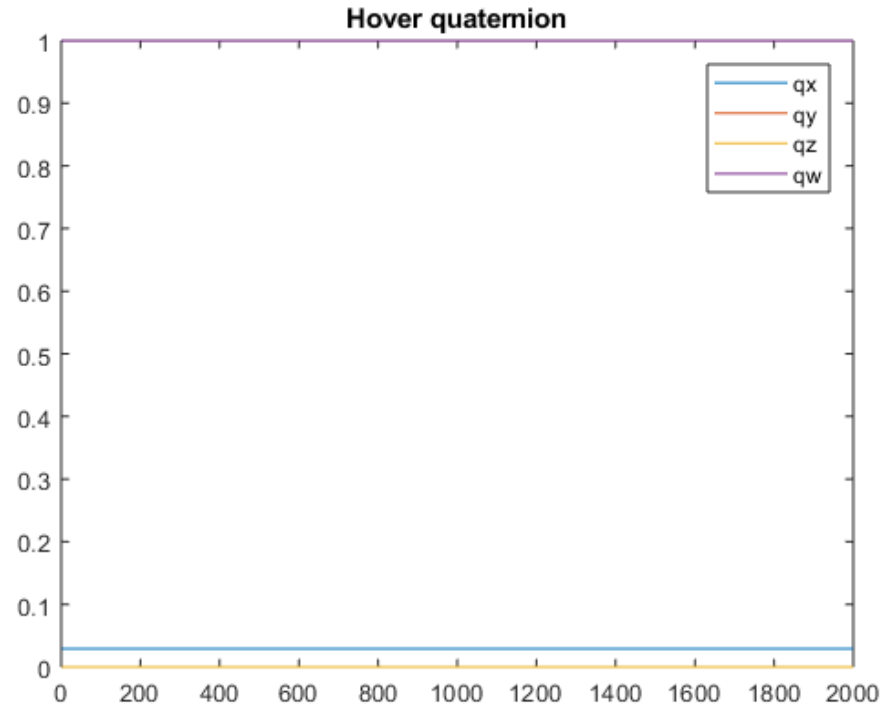

Figure 2: Hover Quaternion without Noise

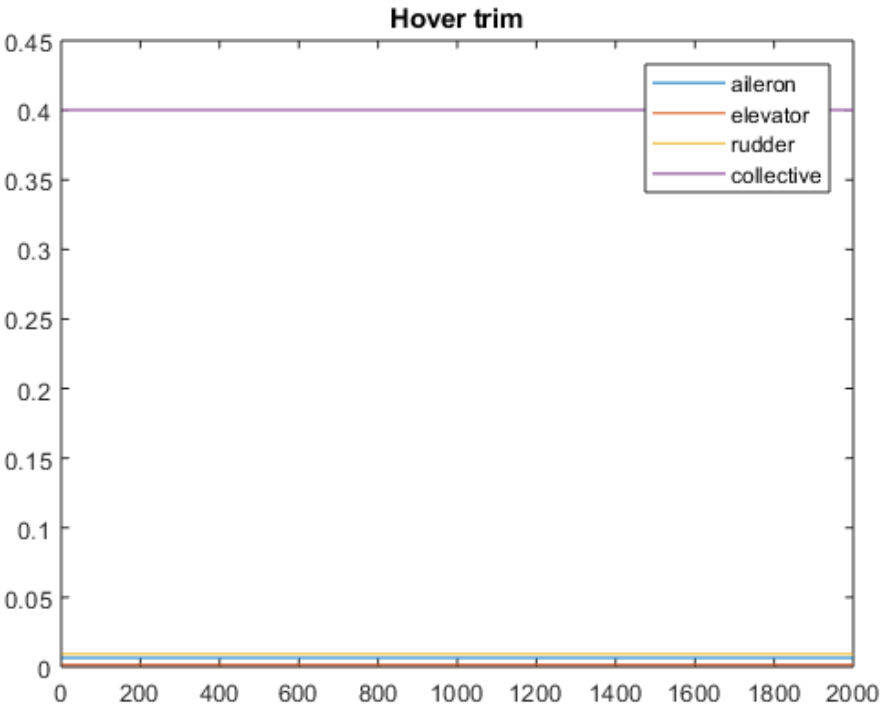

Figure 3: Hover Trim without Noise

For the figure 1, the helicopter is successfully hovering with a very little oscillation (Note that the magnitude of the y-axis is in the order of 10^{-16}, which is negligibly small).

For the figure 2, the helicopter quaternion, which can then be converted to the Euler angles

(Roll ϕ, Pitch θ, Yaw ψ), remains constant and near zero for every time step. This also means that the Euler angles are close to zero and the helicopter is at the level flight.

For the figure 3, the aileron, elevator, and rudder are close to zero, which mean the helicopter does not require much of inputs to translate or rotate in any direction. The reason for the collective not being zero is that the helicopter still requires some amount of thrust to hover (thrust equals to weight).

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.

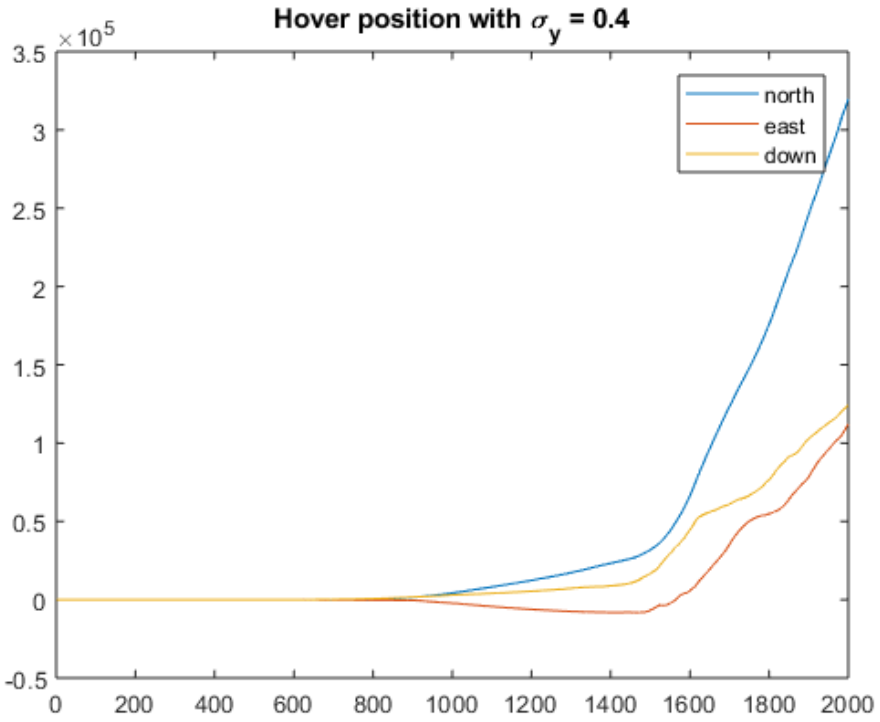

Figure 4: Increasing σ_Y while keeping σ_X zero

The LQR controller fails when there is a noise in the observation because the value of the state corrupted by the noise can be too far off from the operating point and this causes a trouble to the LQR controller.

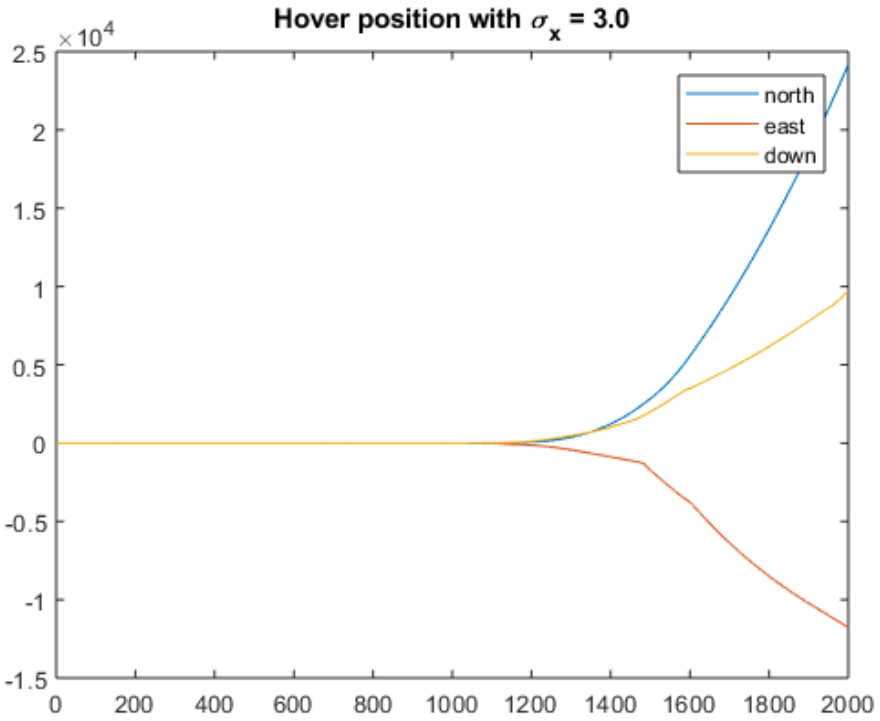

Figure 5: Increasing σ_X while keeping σ_Y zero

The LQR controller fails when there is a noise in the motion model because the LQR gain were calculated from the model without a noise; therefore, when the noise is added this gain is no longer optimal and may not be the stabilizing one, which makes the state blows up.

As we can see from both plots, the system is more sensitive to the observation noise (σ_Y) than the process noise (σ_X). This is because (at least from the given MATLAB code) we are using the full-state feedback ($y = Cx;\ C = I$) i.e. we can observe every aspect of the state including the noise; therefore, the system is less sensitive to the process noise than the observation 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.

The graphs after implementing a Kalman Filter are shown in the following figures.

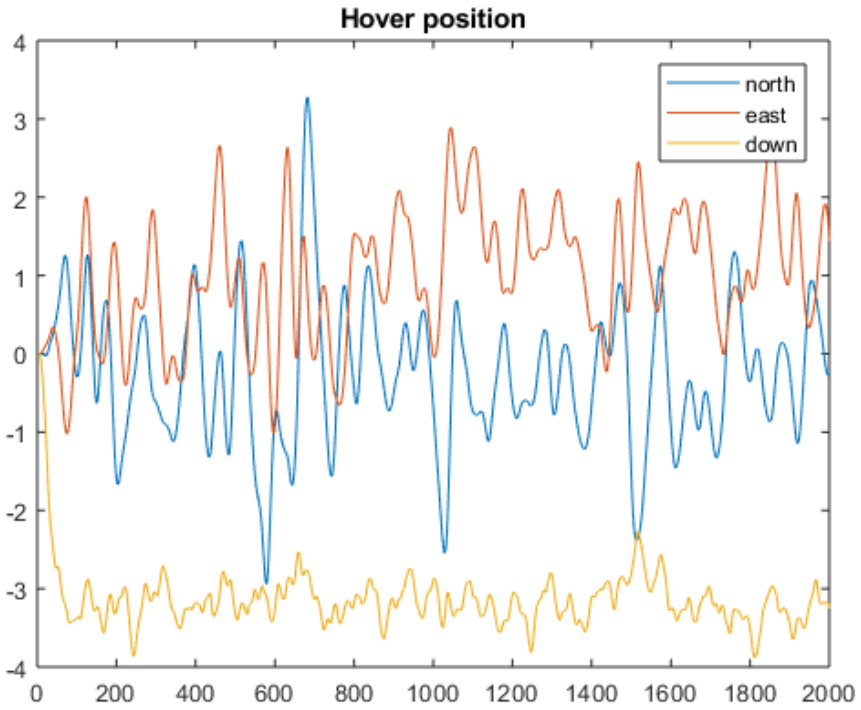

Figure 6: Hover Position with a Kalman Filter

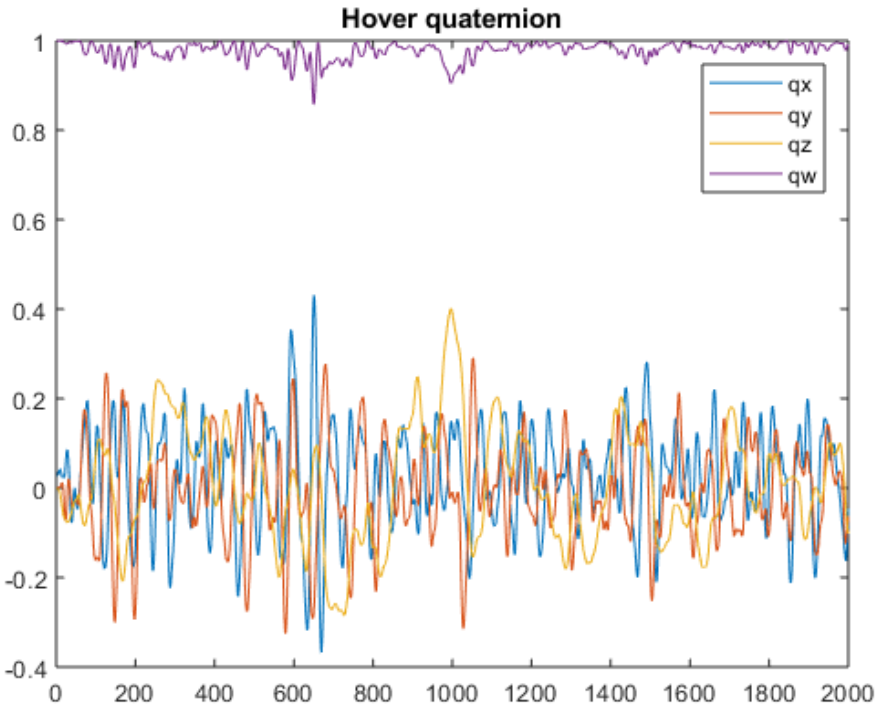

Figure 7: Hover Quaternion with a Kalman Filter

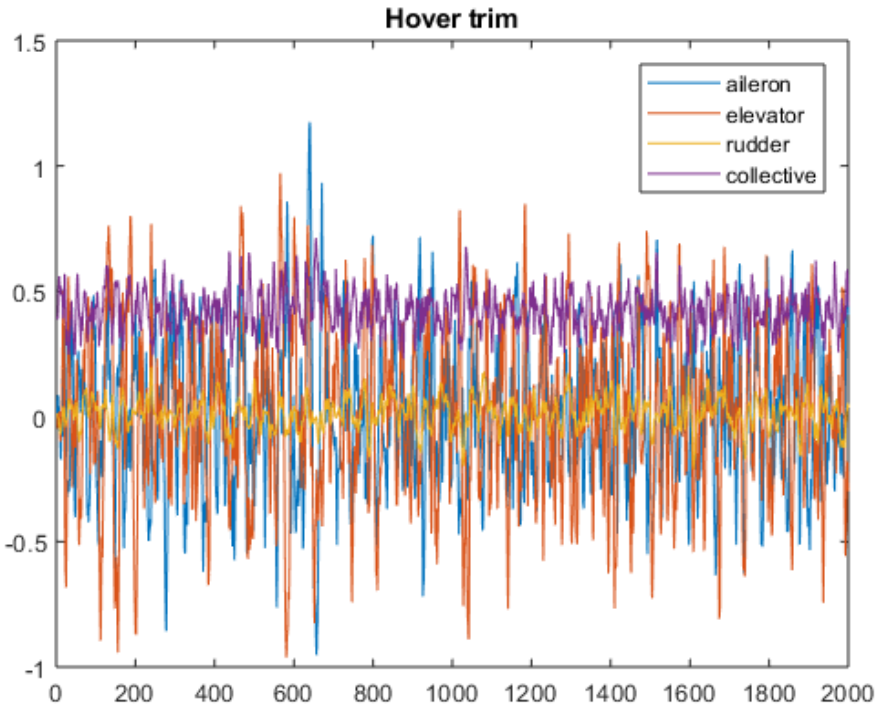

Figure 8: Hover Trim with a Kalman Filter

The parameter Q (process noise covariance matrix), and R (observation noise covariance matrix) are chosen to be $2I$, and $50I$, respectively, where I is an identity matrix.

Also the initial value of covariance matrix P is chosen to be an identity matrix.

The Kalman filter algorithm was implemented as follow.

Algorithm 1 Kalman Filter

Time Update:
$\bar{\mu}_{x,t} \leftarrow A\mu_{x,t-1} + Bu_{t-1}$
$\bar{P} \leftarrow APA^T + Q$

Kalman Gain and Innovation:
$K \leftarrow \bar{P}C^T(C\bar{P}C^T + R)^{-1}$
$\epsilon \leftarrow y_t - C\bar{\mu}_{x,t}$

Measurement Update:
$\mu_{x,t} \leftarrow \bar{\mu}_{x,t} + K\epsilon$
$P \leftarrow (I - KC)\bar{P}$

Although there is an oscillation due to noises, by implementing the Kalman filter, the helicopter is able to hover with a little offset in DOWN-axis. Moreover, the average value of quaternion and trim is close to zero as the first case. (successful hovering without noises).

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

Figure 9: Increasing σ_Y while keeping σ_X zero with KF

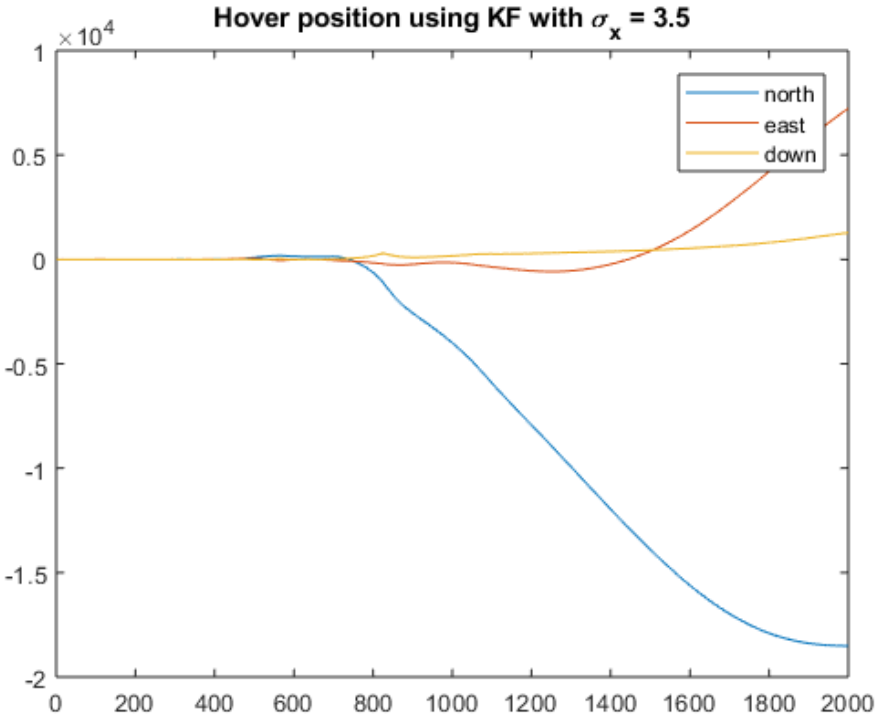

Figure 10: Increasing σ_X while keeping σ_Y zero

When the Kalman filter is implemented the controller is more robust to the noise since both σ_x and σ_y can be increased (before helicopter fails to hover) to a larger number than the case without filtering.

The reason that this case still fails is that the Kalman filter parameters (Q, R) are still the same value before increasing the noise. Therefore, the filtering algorithm cannot well-capture the additional noise and makes the helicopter fails to hover.

5 Implement a non-linear (EKF) Kalman Filter for this system.

The graphs after implementing an Extended Kalman Filter are shown in the following figures. Note that the value of every parameter is the same one used in the Kalman Filter case.

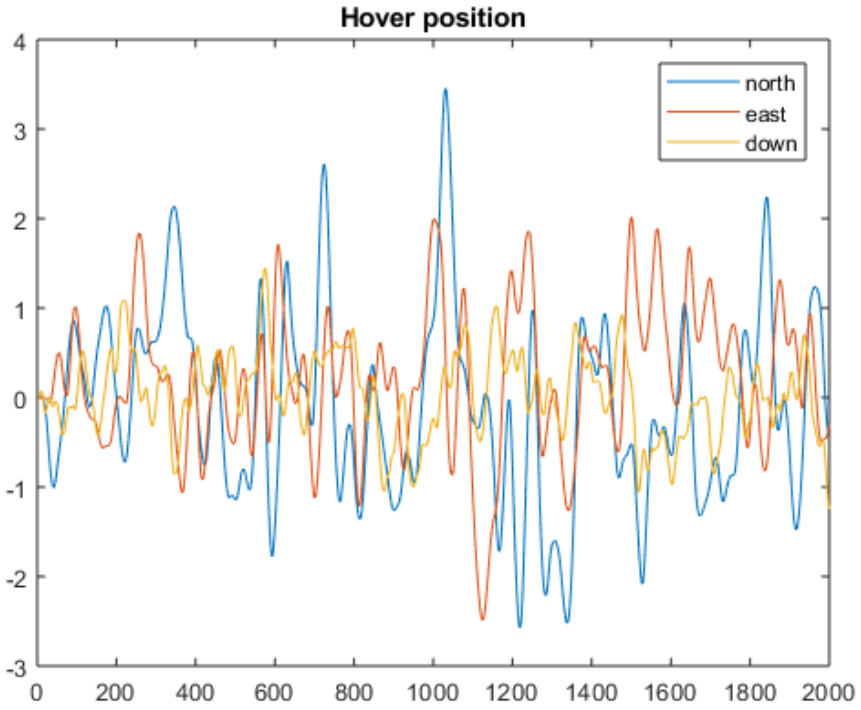

Figure 11: Hover Position with an Extended Kalman Filter

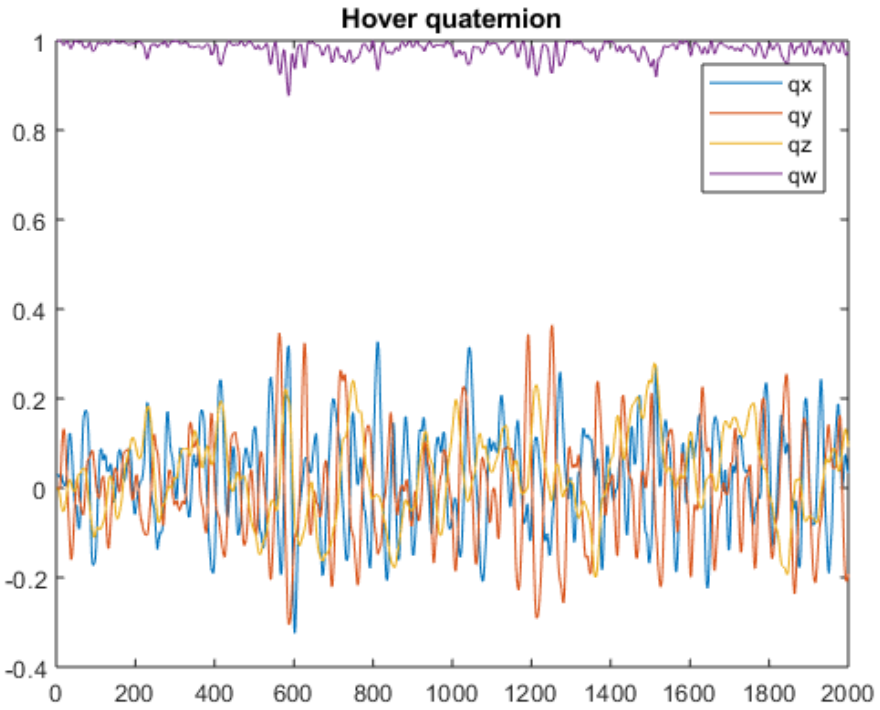

Figure 12: Hover Quaternion with an Extended Kalman Filter

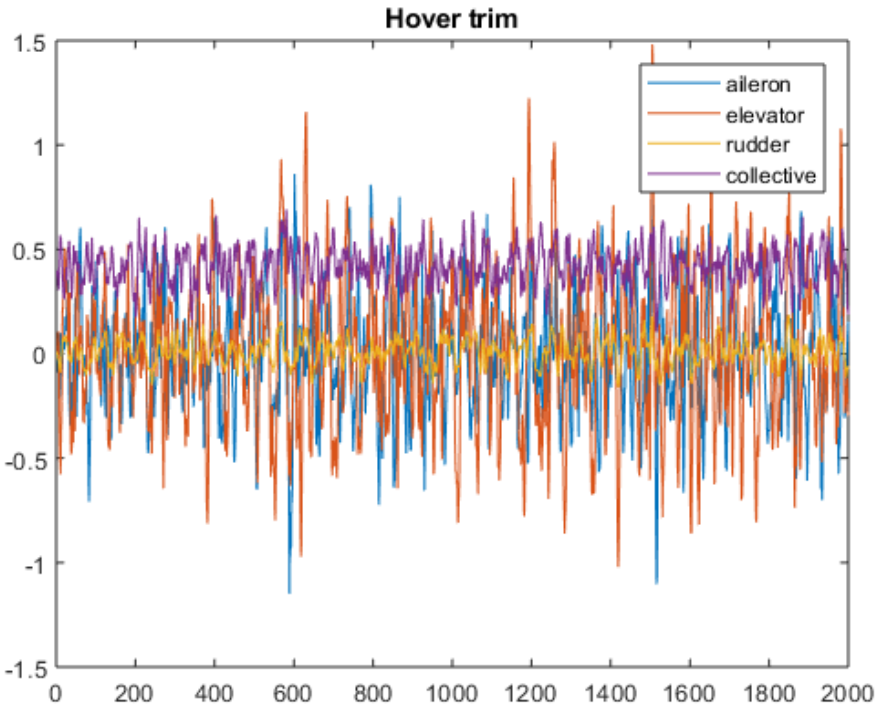

Figure 13: Hover Trim with an Extended Kalman Filter

The Extended Kalman filter algorithm was implemented as follow.

Note that the function $f\_heli()$ and $linearized\_dynamics()$ are given. And since the obser-

vation is full-state $y = g(x) = x \rightarrow C_{EKF} = I$. Also, the noises that are used for $f\_heli()$ are set to be zero because σ_x and σ_y are not supposed to be known since they are external noises e.g. wind, sensor noise.

Algorithm 2 Extended Kalman Filter

Time Update:
$\bar{\mu}_{x,t} \leftarrow f\_heli(\mu_{x,t-1}\ ,\ u_{t-1})$
$[A_{EKF}, B_{EKF}] \leftarrow linearized\_dynamics(\mu_{x,t-1}\ ,\ u_{t-1})$
$\bar{P} \leftarrow A_{EKF} P A_{EKF}^T + Q$

Kalman Gain and Innovation:
$K \leftarrow \bar{P} C_{EKF}^T (C_{EKF} \bar{P} C_{EKF}^T + R)^{-1}$
$\epsilon \leftarrow y_t - C_{EKF} \bar{\mu}_{x,t}$

Measurement Update:
$\mu_{x,t} \leftarrow \bar{\mu}_{x,t} + K\epsilon$
$P \leftarrow (I - KC_{EKF})\bar{P}$

The response behaviors after implementing the Extended Kalman filter are similar to the Kalman filter except the EKF did better in the DOWN-axis since it can eliminate the offset in that axis.

The two following graphs below show the NED graph when both σ_x and σ_y are increased until the helicopter fails to hover.

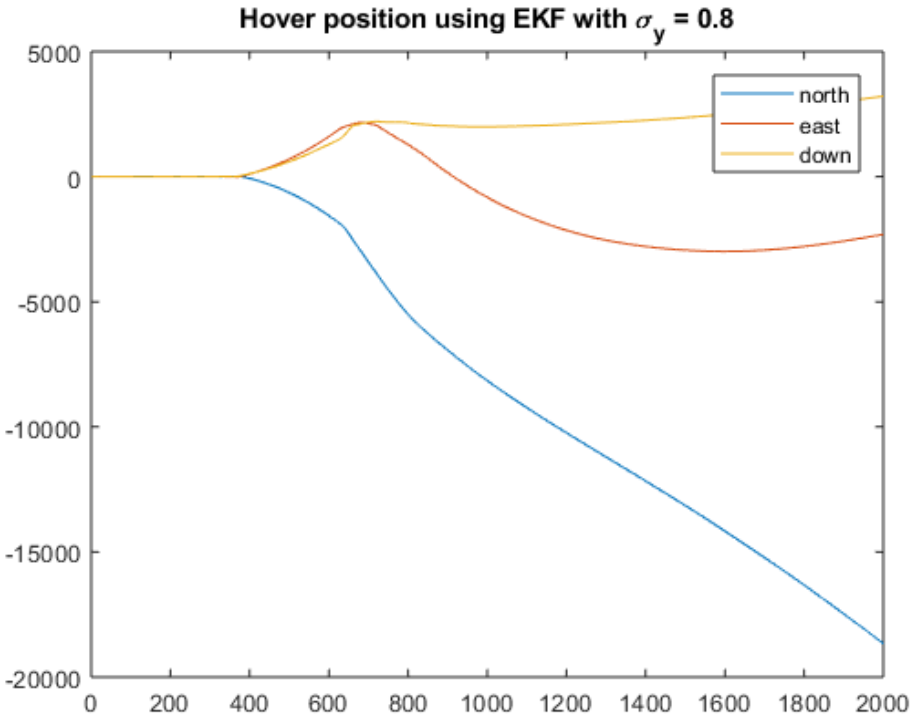

Figure 14: Increasing σ_Y while keeping σ_X zero with EKF

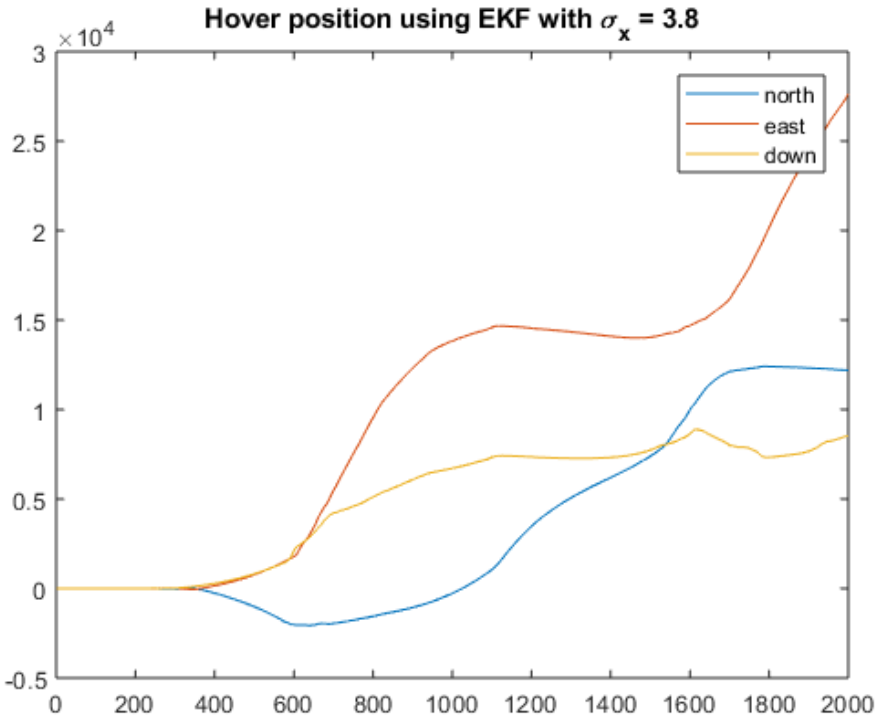

Figure 15: Increasing σ_X while keeping σ_Y zero with EKF

When implementing the Extended Kalman filter, the noise parameters σ_x and σ_y can be increased to a higher number than the cases without filtering and KF because the motion model is more accurate (since using a nonlinear model and performing linearization at every step).

However, this still fails under the same reason as the KF case.

6 MATLAB Code after Implementing KF and EKF

```
% clean up the matlab environment
clear; clc; close all;

% run initialization of some paths and variables
init_setup;
load('lab3.mat');
% contains A, B, C, LQR_Kss, target_hover_state, clipping_distance
% C = eye(size(x)) %full-state

H = 2000;

% actual state, not observable, do not reference
x(:,1) = target_hover_state;

% initial state estimate (given)
mu_x(:,1) = x(:,1);
% initial state observation (given)
y(:,1) = x(:,1);

% initial control
dx = compute_dx(target_hover_state, mu_x(:,1));
u(:,1) = LQR_Kss* dx;
```

```
23
24 % noise parameters X -> Motion model, Y -> Observation
25 sigmaY = 0.5; %default 0.5;
26 %fail at 0.4, 0.7, 0.8 for no filter, KF, EKF, respectively
27
28 sigmaX = 0.1; %default 0.1;
29 %fail at 3.0, 3.5, 3.8 for no filter, KF, EKF, respectively
30
31 % Define Parameters
32 P = 1*eye(size(x,1)); %initial covariance matrix
33 Q = 2*eye(size(x,1)); %process noise matrix
34 R = 50*eye(size(x,1)); %observation noise matrix
35
36 for t=2:H
37         % add noise to motion model:
38     noise_F_T = randn(6,1)*sigmaX;
39
40     % Simulate helicopter, do not change:
41     x(:,t) = f_heli(x(:,t-1), u(:,t-1), dt, model, idx, noise_F_T);
42
43     % add state observation noise
44     v = randn(size(C*x(:,t)))*sigmaY;
45
46     % observe noisy state
47     y(:,t) = C*x(:,t) + v;
48
49     %%%%%%%%%%%%%%%%%%%% Implementing Kalman Filtering
           %%%%%%%%%%%%%%%%%%%%
50
51     % use Kalman filter to calculate mean state
52     % from mu_x(:,t-1), y(t) ,u(t-1)
53     abcdekfghij = 2; % 0=No Filter, 1=KF, 2=EKF
54
55     if abcdekfghij == 0
56       %%%%%%%%%% No Filter %%%%%%%%%%
57         mu_x(:,t) = y(:,t); % dumb take observation as estimate
58
59     elseif abcdekfghij == 1
60       %%%%%%%%%% KF %%%%%%%%%%
61         % Time Update
62         mu_x_bar(:,t) = A*mu_x(:,t-1) + B*u(:,t-1);
63         P_bar = A*P*A' + Q;
64
65         K = P_bar*C'/(C*P_bar*C' + R); % Kalman Gain
66         e = y(:,t) - C*mu_x_bar(:,t); % innovation
67
68         % Measurement Update
69         mu_x(:,t) = mu_x_bar(:,t) + K*e;
70         P = (eye(size(x,1)) - K*C)*P_bar;
71
72     elseif abcdekfghij == 2
73       %%%%%%%%%% EKF %%%%%%%%%%
74         % Time Update
75         mu_x_bar(:,t) = f_heli(mu_x(:,t-1), u(:,t-1), dt, model, idx,
               zeros(6,1)); %E(noise) = 0
76         [A_EKF, ~] = linearized_dynamics(mu_x(:,t-1), u(:,t-1), mu_x(:,
               t-1), mu_x_bar(:,t), @f_heli, dt, model, idx, zeros(6,1),
               0, 0);
```

```
77          P_bar = A_EKF*P*A_EKF' + Q;
78
79          % Since it is a full-state feedback, g(x) = x -> C = I
80          C_EKF = eye(size(x,1));
81          K = P_bar*C_EKF'/(C_EKF*P_bar*C_EKF' + R); % Kalman Gain
82          e = y(:,t) - mu_x_bar(:,t); % innovation
83
84          % Measurement Update
85          mu_x(:,t) = mu_x_bar(:,t) + K*e;
86          P = (eye(size(x,1)) - K*C_EKF)*P_bar;
87      end
88
89      %%%%%%%%%%%%%%%% End of Implementing Kalman Filtering
            %%%%%%%%%%%%%%%%
90
91      % LQR controller generates control for next step.  u(t-1) takes x
            (:,t-1)
92      % to x(:,t).  do not change (only change mu_x value above)
93          dx = compute_dx(target_hover_state, mu_x(:,t));
94      dx(idx.ned) = max(min(dx(idx.ned), clipping_distance),-
            clipping_distance);
95          u(:,t) = LQR_Kss* dx;
96
97  end
98
99  figure; plot(x(idx.ned,:)'); legend('north', 'east', 'down'); title('
        Hover position');
100 figure; plot(x(idx.q,:)'); legend('qx', 'qy', 'qz', 'qw'); title('Hover
         quaternion');
101 figure; plot(x(idx.u_prev,:)'); legend('aileron','elevator','rudder','
        collective'); title('Hover trim');
```

7 Appendix

link to the source code and figures.