CS8803: STR, Lab 2: Kalman Filtering

Yujia Liu

March 2015

1 Objective

We are trying to use a LQR controller to hover the helicopter in a stationary position. Since dynamic model of the helicopter is non-linear, we linearize the model at the given state so that we can take advantage of Kalman filter to give a approximation of the state of the helicopter. And the LQR algorithm would be more efficient with the estimates from the Kalman filter than that simply taken from the observation. The report will include the four tasks proposed in the lab instruction by order.

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

b. 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**.

c. 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**.

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

2 Successful Hover

Setting both error terms (sigmaX, sigmaY in the code) to zero implies that the helicopter's motion will be accurately represented by the motion model, namely the A and B matrix and the helicopter's state will be accurately obtained by the observation model.
The figures of hover position, trim and quaternion are shown in figure 1,2,3. As we can see, the state of the helicopter is stable and steady.

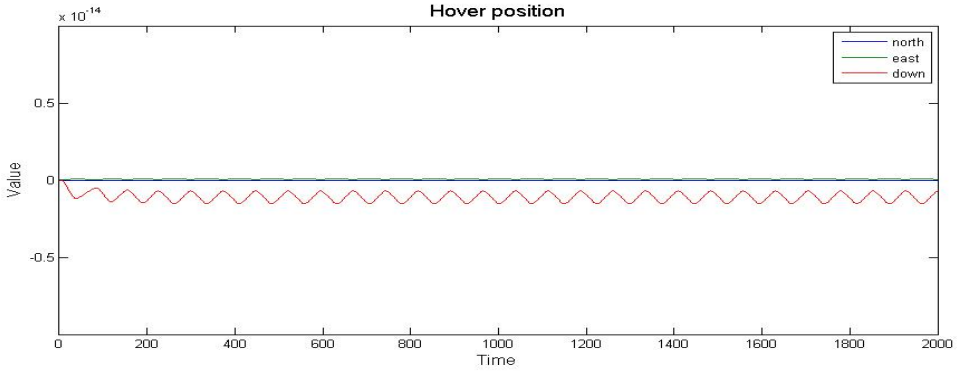

Figure 1

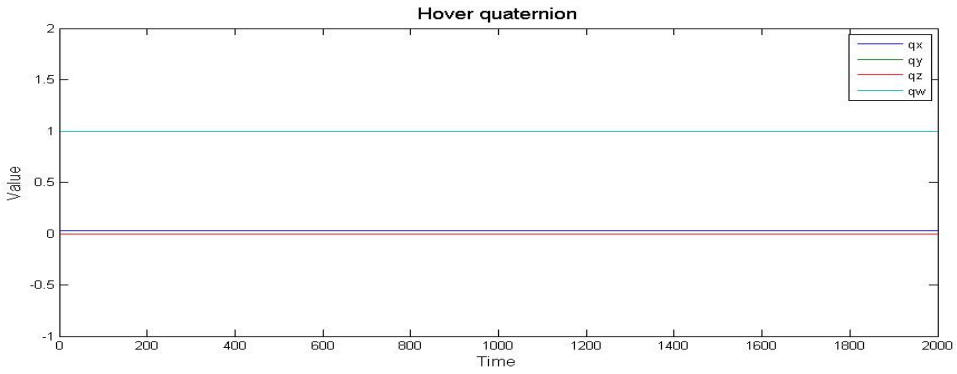

Figure 2

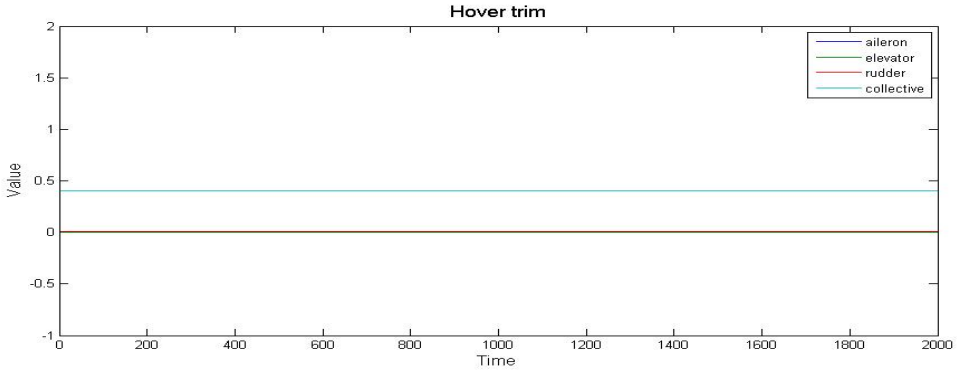

Figure 3

3 Increase Error Until Failure (without Kalman filter)

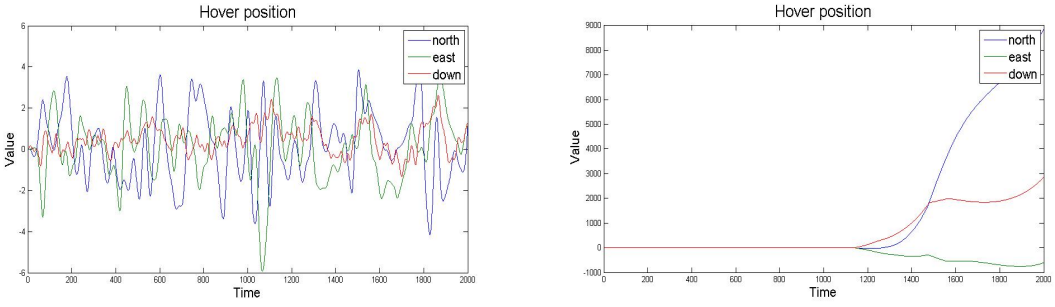

Figure 4

When we increase the error for the observation model, namely the sigmaY, from zero, our estimation of the state from the observation is not absolutely correct any more. When we set sigmaY up to 0.3, the NED (North East Down) graph in two different simulations are shown in Figure 4. We consider it as a successful hover when the position of the helicopter fluctuates in a certain range and a fail hover when it diverges. After multiple times of simulation, we have a conclusion that when sigmaY$\geq$ 0.3, the hover position cannot be maintained every time, i.e., sigmaY = 0.3 is roughly the

threshold for stability without Kalman filter.

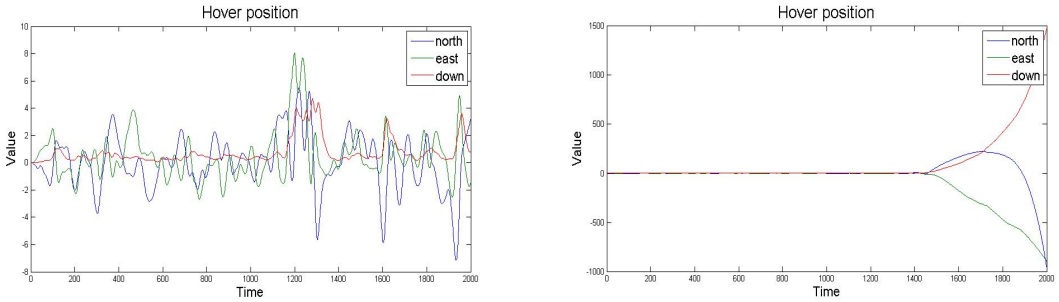

Figure 5

When we increase the error for the motion model, namely the sigmaX, from zero, the helicopter's motion is not absolutely represented by the transition matrices. When we set sigmaX up to 2.5, the NED (North East Down) graph in two different simulations are shown in Figure 5. Same as with sigmaY, we have a conclusion that when sigmaX$\geq$ 2.5, the hover position cannot be maintained every time, i.e., sigmaX = 2.5 is roughly the threshold for stability without Kalman filter.

4 Kalman Filter Implementation

Now we implement Kalman filter to estiamte the state of the helicopter at each time step. When each error is reset to the default values, the helicopter can successfully hover with the help of Kalman filter as shown in figure 6,7,8.

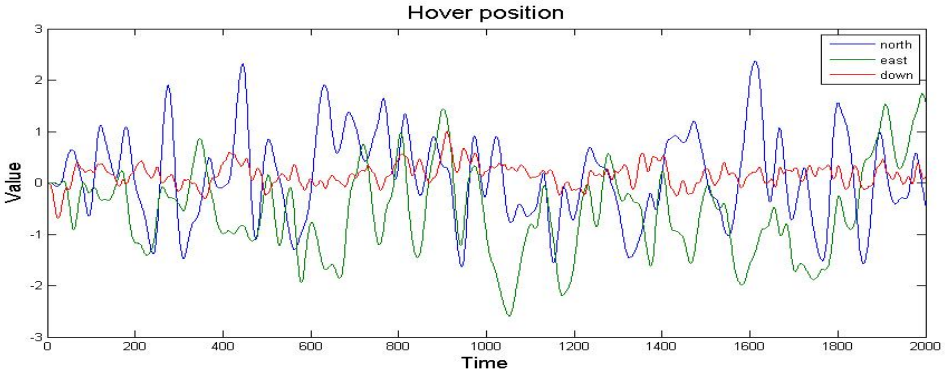

Figure 6

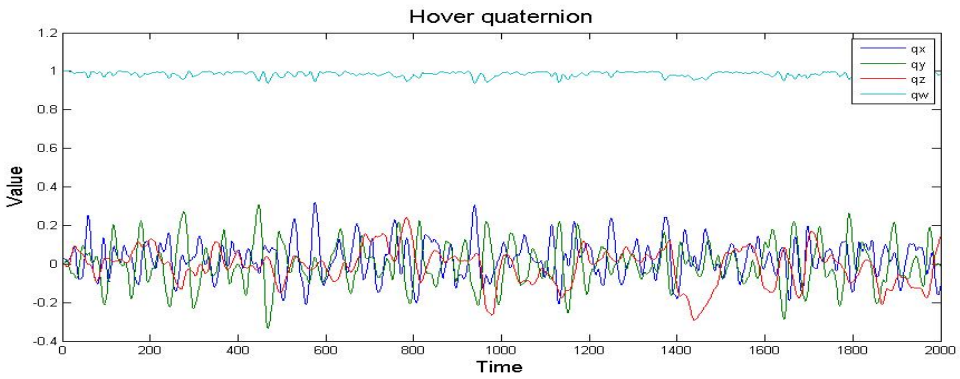

Figure 7

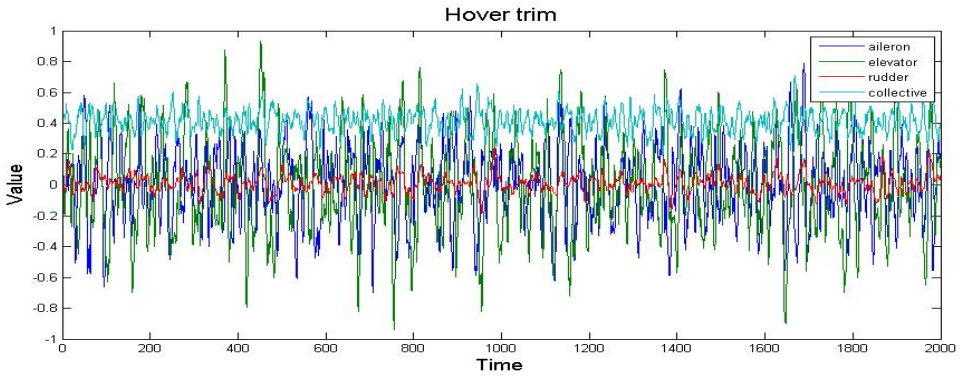

Figure 8

5 Increase Error Until Failure (with Kalman filter)

Now we increase each error, namely the sigmaX and sigmaY, and try to make the Kalman filter collapsed on this system. The parameters sigmaX and sigmaY and their corresponding successful rate is shown in the folowing table.

sigmaX	sigmaY	successful rate
0.1	0.5	1
1	0.5	0.72
1.5	0.5	0.22
2	0.5	0
0.1	0.6	0.66
0.1	0.7	0.09

As we can see, the system is more susceptible to sigmaY than sigmaX. The reason why this system easily diverges is that the Kalman filter is built on the assumption that the system is linear, which is not true for all time. The lineared system can maintain stability with small variation in states. When the states changes by amount larger than some level, the linearization of the system will not hold. The assumption is not true and therefore the Kalman filter is not useful anymore.
In conclusion, the shortcoming of this system is not robust enough for large random variation in motion model and observation model.

Appendix

The Matlab code of Kalman filter part is shown below.

```
% noise parameters
sigmaY = 0.5;
sigmaX = 0.1;

% initial the covariance matrix for state
% and the error covariance matrix for motion and observation model
P(:,:,1) = eye(size(A,1));
Q = eye(size(A,1))*0.1;
R = eye(size(A,1))*5;

for t=2:H
% add noise to motion model:
    noise_F_T = randn(6,1)*sigmaX;

    % Simulate helicopter, do not change:
    x(:,t) = f_heli(x(:,t-1), u(:,t-1), dt, model, idx, noise_F_T);

    % add state observation noise
    v = randn(size(C*x(:,t)))*sigmaY;

    % observe noisy state
    y(:,t) = C*x(:,t) + v;

    % use Kalman filter to calculate mean state
    % from mu_x(:,t-1), y(t) ,u(t-1)
    Mu_temp = A*Mu(:,t-1) + B*u(:,t-1);
    P_temp = A*P(:,:,t-1)*A' + Q;

    offset = zeros(size(A,1),1);
    offset(13) = 1.2;
    offset(14) = -3.6;
    Mu(:,t) = Mu_temp + P_temp*C'/(C*P_temp*C' + R)*(y(:,t) + offset - C*Mu_temp);
    P(:,:,t) = P_temp - P_temp*C'/(C*P_temp*C' + R)*C*P_temp;

    % LQR controller generates control for next step.  u(t-1) takes x(:,t-1)
    % to x(:,t).  do not change (only change mu_x value above)
dx = compute_dx(target_hover_state, mu_x(:,t));
    dx(idx.ned) = max(min(dx(idx.ned), clipping_distance),-clipping_distance);
u(:,t) = LQR_Kss* dx;
```