Lab 4: Kalman Filter

In this lab you will modify the Kalman filter on the drone to try to improve the altitude estimation.

Drone Altitude Kalman Filter

In this lab you will modify the altitude KF for the drone to try to improve the accuracy of the altitude estimate. Currently, the drone altitude KF only uses the sonar and vertical acceleration measurements to estimate the altitude, and does not use the pressure sensor measurement.

The altitude KF is located in the `Drone_Compensator/Estimator/EstimatorAltitude` block in the `sim_quadrotor.slx` Simulink model. It is called `KalmanFilter_altitude` and is pictured in Figure 1.

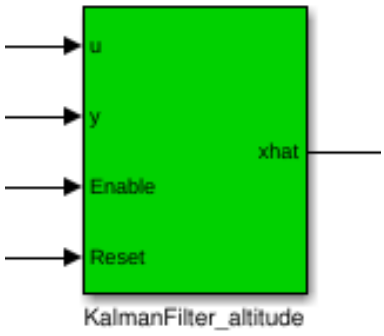

Figure 1: Kalman filter block for altitude estimation.

The input `u` is the vertical acceleration measurement and the measurement `y` is the sonar altitude measurement. The output of the block is the state estimate `xhat`. It is a 2-by-1 vector consisting of the estimate of the altitude and derivative of the altitude.

If the input to the `Enable` port is 1 the Kalman filter is enabled and uses u and y (along with the system dynamics and noise covariance data) to estimate the state. If the input is 0 the Kalman filter is used with only u input and the measurement y .

If the input to the `Reset` port is nonzero, then the Kalman filter will be restarted from the initial conditions set in the block. Otherwise, when it is 0 the Kalman filter will run normally.

Software-in-the-loop simulation

For this lab you will simulate the Kalman filter with data collected from your drone. To do this first copy the `Drone_Compensator` block open into the `sim_SoftwareIntheLoop_Compensator.slx` Simulink

model (this model is located in the `Simulation` folder). Add to this model `To Workspace` blocks (or other Sink blocks) to export the desired data (`states_estimout`, `sensordata_datout`, etc...) to the Matlab workspace.

To run this model with data recorded from the drone first load the `RSdata.mat` file you want to use and run the `FlightAnalyzer.m` script. Now you can run the `sim_SoftwareIntheLoop_Compensator.slx` model and it will simulate the output of the `Drone_Compensator` block with the data recorded from the `RSdata.mat` file.

Problems

Study the Simulink model and answer the following questions:

1. Under what conditions will the `Enable` input be 1?

   Hint: Check the `OutlierHandling` block for two conditions.

2. Under what conditions will the `Reset` input be nonzero?

3. What is the model that the Kalman filter is using? Note that the Matlab notation is slightly different than that used in class. The notation is described in the Matlab help: `https://www.mathworks.com/help/control/ref/kalman.html`

4. Look at the process and measurement noise covariance, Q and R, in the Kalman filter block. How do these values compare to those measured in Lab 2?

5. Modify the altitude reference, the variable `Drone_Compensator_U_orient_pos_refin[2]` in the `rsedu_control.c` file, to make the drone go up and down at different speeds. Test the drone (you can use the PID or LQR controller) and download the data from it. You will use this data in the following problems.

For the following problems you will tune and modify the altitude Kalman filter by simulating it with data you collected in Problem 5. Once you have tuned the Kalman filter to work well on the recorded data you will test it on the drone. For each problem report the process and measurement noise covariance used by the KF, plot the simulated and tested altitude estimate with the altitude reference, pressure, and sonar sensor measurements.

6. Adjust the process and measurement noise covariance on the current KF to see if it can be improved. You may want to start with the variance measurements from Lab 2. Describe the effect of the process and measurement noise covariance on the altitude estimation.

7. Modify the KF so it uses the pressure sensor altitude measurement, instead of the sonar measurement. This sensor is more reliable (i.e. it is not effected by other objects near the drone) and has a larger range, but has much more noise. Therefore, you will have to adjust the measurement covariance to get good results.

8. Modify the KF so that it takes into account both the sonar and pressure measurements. Note that now the C and D matrices will have two rows and the measurement covariance will be a 2-by-2 matrix.

9. Describe which KF you thinks performs the best and why that is the case. The answer to this will depend on the quality of the sensor measurements and environment, so there is not one right answer. For example in some cases the sonar sensor may provide a good enough measurement that adding the pressure measurement does not make any significant difference in the estimate.

10. Now you will write a Kalman Filter block to replace the built-in one in Simulink. In later lab assignments you will modify this block for different systems and to estimate the state of nonlinear systems. There are many different ways to accomplish with but we will use the `MATLAB Function` block with delay blocks to implement the KF as can be seen in Figure 2.

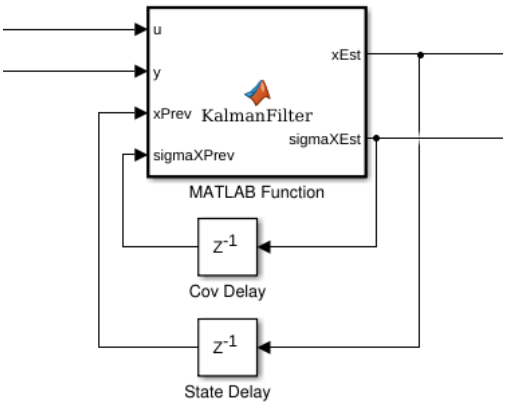

Figure 2: Template for custom Kalman filter.

A template for this Simulink model named `KF_template.slx` can be downloaded from bCourses. This template includes the Matlab function block and delays, but the Matlab function is incomplete. By double clicking on the Matlab function block the corresponding function will open in Matlab as shown in Figure 3.

For whichever variant KF you decided worked the best in Problem 9, program the system dynamics and Kalman filter equations into this function. This function should return the state estimate `xEst` and the state error covariance estimate `sigmaXEst`.

```
function [xEst, sigmaXEst] = KalmanFilter(u, y, xPrev, sigmaXPrev, sigmaW, dT)
%#codegen
% This function implements the discrete time invariant Kalman Filter
%
% Inputs:
%           u - system input
%           y - output measurement
%           xPrev - state estimate at previous time step
%           sigmaXPrev - state covariance estiamte at previous time step
%           sigmaW - noise covariance
%           dT - sampling time
%
% Outputs:
%           xEst - state estimate
%           sigmaXEst - state covariance estimate


% Define System Dynamics
```

Figure 3: Function in the KF block.

The parameter values for the KF need to be added to the `parameters_estimationcontrol.m` file as shown below:

begin code

```
%% KF parameters

% These values are input arguments to the Matlab function
dT = quadEDT.sampletime;
sigmaW = blkdiag(0.0005, 0.1);

% These values are used to initialize the state and covariance delays
x0 = [-0.05 0];
sigmaX0 = zeros(2,2);
```

end code

The parameters `dT` and `sigmaW` are input arguments to the Matlab function and `x0` and `sigmaX0` are the initial conditions for the state and covariance delay blocks. Modify these parameters so they match those used in the built-in KF block.

Plot the altitude estimates from this KF and compare it to the estimates from the built-in KF using the simulated data. How do the estimates compare? If they are significantly different at certain times explain why this might be the case.

Also plot, the entries of the estimated state error covariance matrix. What do you notice about the behavior of these values? Is this behavior expected?