ME 586: Biology-Inspired Robotics

University of Washington, Winter 2020, Prof. Sawyer B. Fuller

Problem Set 3

Goals: familiarize you with basics of adding components in the Robot Operating System for use on the Crazyflie and using a sensor model and Kalman Filter to estimate and control the state of a flying robot in simulation.

Optional Reading

- Optimization-Based Control by Richard M. Murray, Chapter 5: Kalman Filtering.
- Koenderink1987, "Facts on Optic Flow," *Biological Cybernetics* 1987 (on the course web page) presents a derivation of optic flow (slightly different than in class).

I. Robot Operating System on the Crazyflie For this problem you will write three ROS nodes that perform the following functions:

1. A node which accepts keyboard input and publishes that data to a topic.
2. A node which subscribes to the keyboard information and uses that information to control the crazyflie. This node will also publish crazyflie location information to another topic.
3. A node which subscribes to the crazyflie location information and live plots the location.

You will need to look up python libraries to accept keyboard information and to live plot. I recommend the FuncAnimation package from Matplotlib for live plotting. To review ROS nodes and publishing and subscribing, see the lecture notes or go to http://wiki.ros.org/ROS/Tutorials.

To submit: Three `.py` python node files and a screen capture video to show all the parts working together.

II. State estimation using a Kalman Filter The Kalman Filter can estimate the state of the system using sensor readings (typically fewer in number than the size of the state), and a model of the dynamics. Given a dynamical system excited by zero-mean process noise $\boldsymbol{d}$ and measurement noise $\boldsymbol{n}$,

$$\begin{aligned}\dot{\boldsymbol{q}} &= A\boldsymbol{q} + B\boldsymbol{u} + G\boldsymbol{d}\\ \boldsymbol{y} &= C\boldsymbol{q} + \boldsymbol{n}\end{aligned}$$

with covariances $E\{\boldsymbol{d}\boldsymbol{d}^T\} = Q_N = Q_N^T \geq 0$, $E\{\boldsymbol{n}\boldsymbol{n}^T\} = R_N = R_N^T > 0$, a Kalman Filter consists of an observer gain matrix L such that the "state estimation dynamics"

$$\hat{\boldsymbol{q}} = A\hat{\boldsymbol{q}} + B\boldsymbol{u} + L(\boldsymbol{y} - C\hat{\boldsymbol{q}})$$

produces a state estimate $\hat{\boldsymbol{q}}$ that minimizes the expected squared error using the sensor measurements $\boldsymbol{y}$. The quantity $\boldsymbol{y} - C\hat{\boldsymbol{q}}$ represents the error between the current estimate and the sensor readings, which the filter uses to correct its estimate of the full state. Only if the system is *observable* can the full state be estimated.

You will write a Kalman Filter to estimate (but not control) the state of nonlinear downward-hanging pendulum dynamics using a noisy measurement of the angle of the pendulum. The dynamics of the pendulum are given by

$$ml^2\ddot{\theta} = -mg\sin\theta + \tau_d,$$

where m is the mass of at the end of the pendulum, θ is the angle of the pendulum with respect to vertical, and l is the length of the (massless) rod that connects the mass to the pivot at the top. τ_d is a disturbance torque acting on the system. A sensor measures θ according to a measurement $\theta_m = \theta + n_\theta$, where n_θ is a stationary, zero-mean, Gaussian-distributed variable with standard deviation σ_θ.

1. Write the dynamics as a state-space nonlinear function, and provide the A, B, C, and D matrices of the system linearized at $\theta = 0$. Show that the system is observable.

2. To formulate a Kalman Filter, the system must be excited by a non-zero disturbance, and the system must be "controllable" by the noise (i.e. the pair (A, G) must be controllable/reachable). Provide the G matrix corresponding to excitation by a disturbance torque τ_d.

3. Write a simulation of this system's nonlinear dynamics, and implement a Kalman Filter to estimate its state from measurements θ_m (note: this problem entails *estimation*, but not *control*). If you are stuck, it may be helpful to consult the file `me586_example_kalman_estimator.ipynb` on the software_examples section of the course web page for an example implementation of a Kalman Filter on a different system. Please use the following constants:

m	0.1 kg
l	1 m
σ_θ	0.1 rad

   You will also need to add the following function definition into your Jupyter notebook, because it is not yet implemented in the version of the python-control library you downloaded:

```
def lqe(A, G, C, QN, RN):
        'linear quadratic estimator function to calculate Kalman gain'
        A = arr(A, ndmin=2)
        C = arr(C, ndmin=2)
        QN = arr(QN, ndmin=2)
        G = arr(G, ndmin=2)
        P, E, K = control.care(A.T, C.T, G @ QN @ G.T, RN)
        return K.T, P, E
```

   Note: estimating the standard deviation σ_d of the disturbance noise τ_d is difficult in practice, so in this problem you will instead use it as a tuning knob to achieve desired performance.

4. For an initial angle of $\theta = 1$ rad, your Kalman filter should follow the true state of the pendulum relatively well. Provide a plot in your Jupyter notebook. Then, in a subsequent cell, re-simulate the system with an initial condition of $\theta = 2$ rad. Provide a plot that compares that estimate to an estimate using the linearized (A-matrix) model of your system.

5. Why does a larger excursion result in larger error in your Kalman Filter estimate?

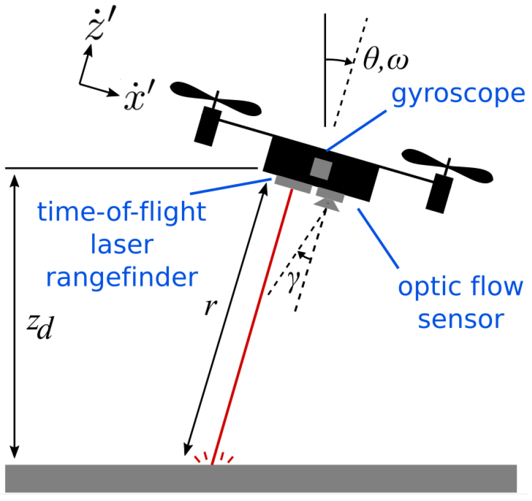

Figure 1: Sensors of the quad-rotor Crazyflie with the "Flowdeck" sensor.

III. State estimation and LQG control of a 2D simulated quadrotor A complete Linear Quadratic Gaussian ("LQG") control system consists of an LQR controller and a "Kalman Filter." You will implement an LQG controller for 2D quadrotor dynamics for a Crazyflie equipped with a "flowdeck" sensor board.

To implement a Kalman Filter, we must first ascertain whether our system is observable at equilibrium. To do that we must linearize the observation model. The onboard gyroscope measures angular velocity, providing a measurement ω_m, the time-of-flight sensor provides a measurement the distance to the surface below the helicopter r_m, and the optic flow sensor provides an optic flow measurement Ω_m , according to the following models

$$\begin{aligned}\omega_m &= \omega + n_g \\ r_m &= r + n_t \\ \Omega_m &= \Omega + n_o = -\omega + \frac{1}{r}(\dot{x}' \cos\gamma - \dot{z}' \sin\gamma) + n_o,\end{aligned}$$

where each n corresponds to noise added to the sensor reading, which we assume is zero-mean Gaussian white noise, and γ is the angle at which optic flow is being measured. $\dot{x}'$ and $\dot{y}'$ are velocities given in body-attached coordinates that are components of the velocity vector $\boldsymbol{v}' = R^T\boldsymbol{v}$. A real optic flow sensor averages the optic flow reading across its field of view, that is, $\Omega_m = \frac{1}{\gamma_2-\gamma_1}\int_{\gamma_1}^{\gamma_2}\Omega(\gamma)d\gamma$ but assume for this exercise that it is simply measuring the optic flow at the angle $\gamma = 0$. These quantities are depicted in Figure 1.

1. For a downward-oriented optic flow sensor ($\gamma = 0$), find the optic flow as a function of your state $\Omega = \Omega(\theta_y, \omega_y, x, \dot{x}, z, \dot{z})$. To calculate this, you will need to find a relation for the distance r to the ground (as a function of θ_y and z) as shown in Figure 1.

2. Now suppose the robot is hovering near equilibrium near the desired height z_d, that is, $\boldsymbol{q} = [\theta_y, \omega_y, x, \dot{x}, z, \dot{z}]^T = [0, 0, 0, 0, z_d, 0]^T$. Show that the linearization of the nonlinear observation model $\boldsymbol{y}(\boldsymbol{q}) = [\omega, r, \Omega]^T$ at equilibrium is given by the observation matrix

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & \frac{1}{z_d} & 0 & 0 \end{bmatrix}$$

3. Show in your notebook, by computing the observability matrix, that with the A matrix derived in the previous problem set, that such a system is *not observable.*

4. The reason for this is that the optic flow sensor does not measure position, only velocity, meaning that the state x is not technically observable. But the math that calculates the gains L of a Kalman filter require a system that is observable. We would like to nevertheless use feedback to control our position relative to our "best guess" of position. We will create a "reduced" linearized model of our system that does not include its non-observable x-position. **Show that the pair (A_r, C_r) of these reduced matrices are observable, and are given by**

$$A_r = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad C_r = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & \frac{1}{z_d} & 0 & 0 \end{bmatrix}.$$

5. The matrix G specifies how disturbance noise $\boldsymbol{d}$ enters the system. This can be set to the identity matrix, but it typically is set to represent a realistic model for what noise might enter the system. The only requirement is that the noise must be able to "control" the system, i.e, the pair (A_r, G_r) must be controllable. Here, we will assume that there are three sources of disturbance: a) rotational torque around the y-axis, b) force in the x-direction and c) force in the z-direction. Show that if the

disturbance is specified in units of [Nm] for torque and [N] for force, then G_r is given by

$$G_r = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{J_{yy}} \\ \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 \end{bmatrix}.$$

6. In the Jupyter notebook `me586_2d_crazeflie_simulator_LQG.ipynb` on the software_examples section of the course webpage, we will use an estimator that is augmented by adding a row of zeros to the L matrix on the row corresponding to the x position state in the full system. This entails no correction of the position estimate $\hat{x}$, but still allows the lateral velocity estimate derived from the optic flow sensor to be numerically integrated, providing an estimate of position that slowly drifts due to sensor noise. Finish implementation of the Kalman Filter (LQE estimator) by filling in missing elements (between comments with three `###`'s.) You will need to add an extra row (in the right location) to your G_r matrix as well. A suggested LQR controller is also implemented that does not need to be changed. As in the previous problem, the size of the sensor noise is known but the disturbance noise size is much harder to measure and therefore becomes our tuning knob. Explore how changing weights affects the behavior of the system, increasing or decreasing the weights in the process noise matrix Q_N until you get reasonable estimator performance. Provide a plot in your notebook that shows the system estimating the state of the system under control of an LQR controller that is operating using feedback from the *true state* rather than the *estimated state* (this is usually easier to get to work correctly).

7. Next, close the loop, giving your LQR controller the Kalman Estimate, and provide a plot showing your results.
Tip: The Kalman filter has a lot of moving parts and can show erratic behavior if gains are too high. One thing you can do to help isolate such problems is to reduce the time increment of your simulation to get a more precise dynamics simulation.

8. Lastly, provide plots that show your system subject to the following effects and briefly (in a sentence or two) explain:

   (a) What happens when the disturbance noise in the x-direction is much higher than the disturbance noise in the z-direction?

   (b) Why does the position estimate have an error if the altitude is not exactly equal to z_d? (such as if you start above or below z_d) Can you suggest an approach you could use to fix it?