AE483 Lab Manual: Week #3
Parameter Estimation and Pitch Control

T. Bretl

September 16, 2019

1 Goal

Your goal in the next two weeks will be to design, implement, and test a controller that makes the pitch angle of the drone achieve a desired value, and to characterize the closed-loop frequency response of this controller. During this first week, you will do the following things:

- Estimate the parameters that relate motor commands, spin rates, and applied forces and torques due to rotors (Section 2).
- Estimate all other unknown parameters in your equations of motion (Section 3).
- Program the drone to accept user-defined parameters (Section 4).
- Derive equations of motion for the drone when mounted on the pitch test stand (Section 5).

Section 6 has a summary of in-lab deliverables. **Make sure to save all data for later analysis.**

2 Estimate rotor parameters

2.1 Estimate k_F with data collected from the motor force measurement rig

Your TA will use LabView to collect data from a motor force measurement rig (Figure 1). This rig sends a command to a single motor/rotor and measures both the spin rate (in revolutions per second) and the aerodynamic force along the axis of rotation (in Newtons). The LabView code will cycle through a number of different motor commands, each applied for about the same length of time. The raw data you receive will have file extension `.lvm` and will look something like this:

```
LabVIEW Measurement
Writer_Version  2
Reader_Version  2
Separator       Tab
Decimal_Separator       .
Multi_Headings  Yes
X_Columns       Multi
Time_Pref       Relative
Operator        gfalcon2
Date    2019/09/10
Time    16:43:39.4628968238830566406
***End_of_Header***

Channels        3
Samples 63000           1876            1876
Date    2019/09/10              2019/09/10              2019/09/10
Time    16:43:39.4628968238830566406            16:43:39.4628968238830566406            16:43:39.4628968238830566406
X_Dimension     Time            Time            Time
X0      0.0000000000000000E+0           0.0000000000000000E+0           0.0000000000000000E+0
Delta_X 0.001000                1.000000                1.000000
***End_of_Header***
X_Value Collected       X_Value Untitled        X_Value Untitled 1      Comment
```

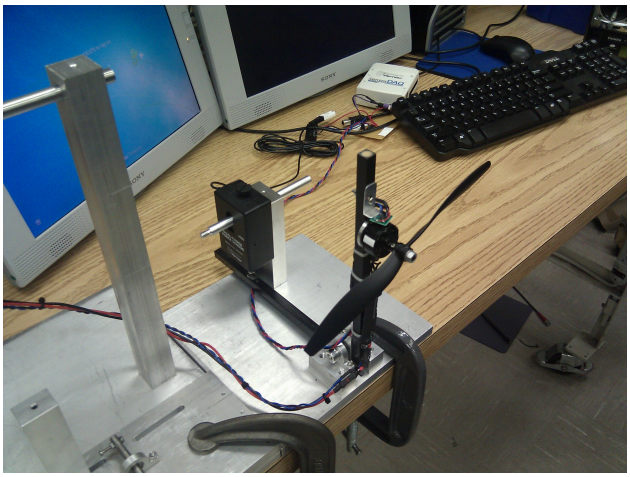

Figure 1: Motor force measurement rig.

```
0.000000    −0.449994    0.000000    0.290000    0.000000    23.524845
0.001000    −0.453262    1.000000    0.846000    1.000000    23.508300
0.002000    −0.453262    2.000000    0.931000    2.000000    23.513759
0.003000    −0.443459    3.000000    1.009000    3.000000    23.527059
0.004000    −0.440191    4.000000    1.090000    4.000000    23.549036
0.005000    −0.459798    5.000000    1.175000    5.000000    23.561844
0.006000    −0.436923    6.000000    1.257000    6.000000    24.082217
0.007000    −0.446727    7.000000    1.338000    7.000000    27.994247
...
...
```

There is a header (lines 1-22) followed by many rows of measurements, each with up to six columns. Columns 1 and 2 tell you the force—the entry in column 2 is the force measurement, and the entry in column 1 is the time at which this force measurement was taken. Columns 4 and 6 tell you the spin rate—the entry in column 6 is the spin rate measurement, and the entry in column 4 is the time at which this spin rate measurement was taken. Notice that the force measurements and the spin rate measurements are, in general, *not* taken at the same time. Moreover, you will likely find that there are many more force measurements than spin rate measurements (although both sets of measurements will span the entire range of times in the experiment). So, you'll want to take care to align these measurements in time before proceeding with analysis.[1] One more thing to note is that all of the force measurements have a bias (or offset) of -0.442 N, which you'll have to correct before proceeding. Once you have parsed the data and corrected the bias, the result will look similar to what is shown in Figure 2. As you know, the relationship between the spin rate σ and the aerodynamic force f along the axis of rotation is

$$f = k_F \sigma^2$$

for some constant k_F. Estimate k_F, given your measurements of σ and f. (There are many different ways of doing this. You may want to start by plotting measurements from both sensors on the same axis, demonstrating your ability to align these two sources of data in time.)

2.2 Estimate k_M with data collected from the motor torque measurement rig

Your TA will also use LabView to collect data from a motor torque measurement rig. This rig works exactly like the motor force measurement rig, but—through the use of a lever arm—converts the applied torque from the rotor into a force that can be measured by a force sensor. So, in addition

[1] In MATLAB, the function `interp1` may be useful for this purpose. Many other methods are also possible.

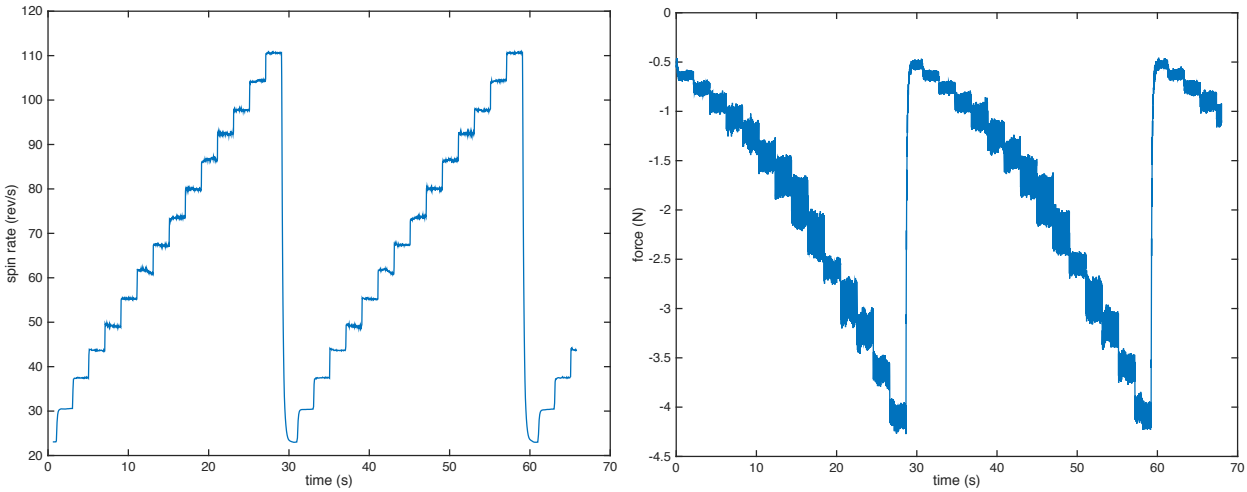

Figure 2: Example data collected from the motor force measurement rig.

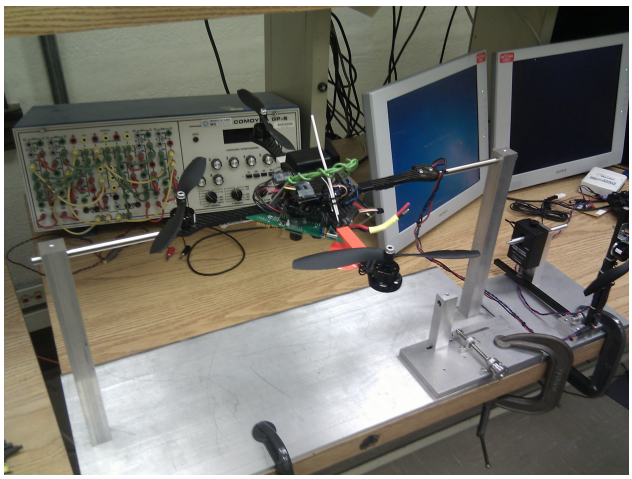

Figure 3: Pitch axis test stand.

to correcting for the bias (-0.442 N) of the force measurements, you'll also need to estimate the length of the lever arm (e.g., with a ruler) and use it to convert the force measurements to torque measurements. Again, the relationship between the spin rate σ and the aerodynamic torque τ about the axis of rotation is

$$\tau = k_M \sigma^2$$

for some constant k_M. Estimate k_M, given your measurements of σ and τ.

2.3 Estimate relationship between motor command and spin rate

In the on-board code, you don't actually get to specify the spin rate directly—instead, you specify a "motor command" that is converted to a desired spin rate by the electronic speed controllers (ESCs) that are in the drone. This motor command is between 0 (minimum speed) and 200 (maximum speed). Lines 219-232 of `sdk.h` in the SDK tell you what the relationship between motor command and spin rate is supposed to be:

```
219 /*
220  * Quadcopter (AscTec Hummingbird, AscTec Pelican -> both Hacker motors)
221  *
222  * motor commands equal a real commanded rpm of (25+(motor*175)/200)*43
223  * please note that all fractions are removed during every calculation steps
224  * e.g. motor = 100; means a turning speed of (25+ (100*175/200))*43=4816 rpm
225  * Max. turning speed 8600 rpm
226  *
227  * motor[0]: front
228  * motor[1]: rear
229  * motor[2]: left
230  * motor[3]: right
231  *
232  */
```

It is important to verify this relationship. It turns out you can do so with the same data you used to compute k_F and k_M. In particular, here is the on-board code in `lab.c` that was running during the motor calibration experiments:

```
1  void lab(void) {
2      motor_rpm_calibration();
3  }
4
5  void motor_rpm_calibration(void) {
6      static int timer=0;
7
8      WO_SDK.ctrl_mode=0x00;  //0x00: direct individual motor control: individual commands for
        motors 0..3
9                              //0x01: direct motor control using standard output mapping:
       commands are interpreted as pitch, roll, yaw and thrust inputs; no attitude controller
       active
10                             //0x02: attitude and throttle control: commands are input for
       standard attitude controller
11                             //0x03: GPS waypoint control
12
13     WO_SDK.ctrl_enabled=1;  //0: disable control by HL processor
```

```
                            //1: enable control by HL processor

    WO_SDK.disable_motor_onoff_by_stick=0;

    //scale throttle stick to [0..200] and map it to all motors
    WO_Direct_Individual_Motor_Control.motor[0]=0;
    WO_Direct_Individual_Motor_Control.motor[1]=0;

    timer++;
    if(timer<2000) {
        WO_Direct_Individual_Motor_Control.motor[2]=10;
        WO_Direct_Individual_Motor_Control.motor[3]=10;
    }else if(timer<4000) {
        WO_Direct_Individual_Motor_Control.motor[2]=20;
        WO_Direct_Individual_Motor_Control.motor[3]=20;
    }else if(timer<6000) {
        WO_Direct_Individual_Motor_Control.motor[2]=30;
        WO_Direct_Individual_Motor_Control.motor[3]=30;
    }else if(timer<8000) {
        WO_Direct_Individual_Motor_Control.motor[2]=40;
        WO_Direct_Individual_Motor_Control.motor[3]=40;
    }else if(timer<10000) {
        WO_Direct_Individual_Motor_Control.motor[2]=50;
        WO_Direct_Individual_Motor_Control.motor[3]=50;
    }else if(timer<12000) {
        WO_Direct_Individual_Motor_Control.motor[2]=60;
        WO_Direct_Individual_Motor_Control.motor[3]=60;
    }else if(timer<14000) {
        WO_Direct_Individual_Motor_Control.motor[2]=70;
        WO_Direct_Individual_Motor_Control.motor[3]=70;
    }else if(timer<16000) {
        WO_Direct_Individual_Motor_Control.motor[2]=80;
        WO_Direct_Individual_Motor_Control.motor[3]=80;
    }else if(timer<18000) {
        WO_Direct_Individual_Motor_Control.motor[2]=90;
        WO_Direct_Individual_Motor_Control.motor[3]=90;
    }else if(timer<20000) {
        WO_Direct_Individual_Motor_Control.motor[2]=100;
        WO_Direct_Individual_Motor_Control.motor[3]=100;
    }else if(timer<22000) {
        WO_Direct_Individual_Motor_Control.motor[2]=110;
        WO_Direct_Individual_Motor_Control.motor[3]=110;
    }else if(timer<24000) {
        WO_Direct_Individual_Motor_Control.motor[2]=120;
        WO_Direct_Individual_Motor_Control.motor[3]=120;
    }else if(timer<26000) {
        WO_Direct_Individual_Motor_Control.motor[2]=130;
        WO_Direct_Individual_Motor_Control.motor[3]=130;
    }else if(timer<28000) {
        WO_Direct_Individual_Motor_Control.motor[2]=140;
        WO_Direct_Individual_Motor_Control.motor[3]=140;
    }else if(timer<30000) {
        WO_Direct_Individual_Motor_Control.motor[2]=150;
        WO_Direct_Individual_Motor_Control.motor[3]=150;
```

```
68      }else {
69          timer=0;
70      }
71  }
```

This code starts with a motor command of 10 and, after each two second (i.e., 2000 millisecond) interval, increases this command by +10, up to 150. This is why your plots of spin rate look like stair steps. In particular, you should be able to create a third set of measurements (for both the force rig data and the torque rig data), which specify the motor command as a function of time. Use this to estimate the relationship between motor command μ and spin rate σ—it will likely have the form

$$\sigma = \alpha\mu + \beta$$

for some constants α and β. Compare the constants that you obtain with those described in `sdk.h`. (Don't forget to express everything to the same units!)

3 Estimate all other parameters

3.1 Find the center of mass

Use any method you like to find the center of mass of the drone, and mark its location in some way.

3.2 Estimate ℓ by measuring the spar length with a ruler

Use a ruler to measure the distance ℓ from the rotation axis of a single rotor to the center of mass. Verify that all four rotors are approximately the same distance from the center of mass.

3.3 Estimate m by measuring the mass of the quadrotor with a scale

Use a scale to measure the mass m of the quadrotor. You may find that you have to measure the mass of the battery and the mass of the quadrotor without the battery separately, and add them (the range of the scale may not be enough to measure both at once).

3.4 Estimate moments of inertia

We assume that the moment of inertia matrix J is diagonal, when written in the body frame. So, it has the form

$$J = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix}$$

for principal moments of inertia J_1 (about the x or roll axis of the body frame), J_2 (about the y or pitch axis of the body frame), and J_3 (about the z or yaw axis of the body frame). Although you should find (see Section 5) that the equations of motion for the drone on the pitch test stand depend only on J_2, it will be important later to have estimates of all three principal moments of inertia. Here is one way that you might measure the moment of inertia about the pitch axis (a similar approach could be used to measure the moment of inertia about other axes):

- Suspend the drone with a wire tied to the end of the $+x$ axis.
- Holding the end of the wire at a fixed position, displace the drone slightly along the $+z$ axis.

- Let the drone swing from the wire and measure the period of oscillation (e.g., use a stopwatch to find how long it takes to swing back and forth some number of times).
- Apply the compound pendulum formula[2] to solve for the moment of inertia from the period.

You are encouraged to find ways of improving this approach (e.g., by suspension with more than one wire) and are welcome to discuss it among all groups in your lab section. Note that, if you wanted to, you could run the on-board and off-board code to collect measurements with the ground station, perhaps getting a better estimate of the oscillation period than you could have obtained by eye with a stopwatch. You are also encouraged to find a completely different way of estimating the moments of inertia, if you like—a variety of different approaches are possible.

4 Program the drone to accept user-defined parameters

Follow the instructions in Appendix E to add at least one user-defined parameter to your on-board code. Show your TA that you can set the value of this parameter using the ACI Tool.[3]

5 Derive equations of motion

5.1 Forces and torques from spin rates

Only the front (along the positive x_{body} axis) and rear (along the negative x_{body} axis) rotors spin when the drone is on the pitch test stand—the other rotors are detached from the drone. Suppose:

$$\sigma_1 = \text{the spin rate of the front rotor (in rad/s)}$$
$$\sigma_2 = \text{the spin rate of the rear rotor (in rad/s).}$$

Write an expression for:

- the force f_{rotors} due to these two rotors in a direction opposite to the gravity vector (i.e., in the $-z_{\text{room}}$ direction), and
- the torque τ_{rotors} due to these two rotors about the pitch axis

in terms of σ_1 and σ_2, the spar length ℓ, the force constant k_F, and the torque constant k_M.

5.2 Spin rates from forces and torques

If the drone were *not* mounted to the pitch test stand, the force f_{rotors} generated by the rotors in a direction opposite to the gravity vector would have to overcome the force of gravity in order for the drone to remain at a constant altitude. Write an expression for the spin rates σ_1 and σ_2 that would overcome gravity (i.e., that would maintain $f_{\text{rotors}} = mg$) while achieving an arbitrary pitch torque τ_{rotors}.

[2] See wikipedia, for example.

[3] You might wonder if the parameter value has *really* been changed, even if ACI Tool is telling you that it has. You have the power to find this out. In particular, you could publish the same variable *both* using `aciPublishVariable` and `aciPublishParameter`, and could modify your ground-station code to read and print the value of this variable to the data file. This would give me a lot more confidence that its value has actually been changed by the ACI Tool.

5.3 Motor commands from spin rates

Using your result from Section 2.3, write an expression for the motor commands μ_1 and μ_2 that would achieve arbitrary spin rates σ_1 and σ_2, respectively.

5.4 Ordinary differential equations (ODEs)

We derived the following equations of motion for a quadrotor:

$$\begin{aligned} \dot{o} &= v \\ \begin{bmatrix} \dot{\theta}_{\text{yaw}} \\ \dot{\theta}_{\text{pitch}} \\ \dot{\theta}_{\text{roll}} \end{bmatrix} &= Nw \\ \dot{v} &= (1/m)f \\ \dot{w} &= J^{-1}\left(\tau - \widehat{w}Jw\right), \end{aligned}$$

where f is the sum of applied forces in the room frame and τ is the sum of applied torques in the body frame. When the quadrotor is on the pitch test stand, we can assume that it does not translate and that it does not rotate about the yaw or roll axis:

$$v_1 = v_2 = v_3 = 0 \qquad \theta_{\text{yaw}} = \theta_{\text{roll}} = 0 \qquad w_1 = w_3 = 0.$$

Write the remaining two ordinary differential equations that describe motion about the pitch axis in terms of the torque τ_{rotors} due to the front and rear rotors about the pitch axis.

6 Summary of in-lab deliverables

You should have done the following things in lab:

1. Show the TA your estimates of k_F, of k_M, and of the constants (e.g., α and β) that govern the relationship between motor command and spin rate (Section 2).
2. Show the TA your estimates of ℓ, m, and the principal moments of inertia (Section 3).
3. Show the TA that you can modify at least one parameter in your on-board code using the ACI Tool (Section 4).
4. Show the TA your equations of motion, as well as your formulae to convert forces and torques to spin rates, and then to motor commands (Section 5).