AE 483
UAV Navigation and Control
Lab Manual

Lab 2

Parameter Estimation and Introduction to Control

2.1 Objectives

In this lab, you will estimate parameters in the hummingbird equations of motion. You will also develop a regulating controller for a one degree-of-freedom test stand. In particular, you will

- Measure the physical properties of the hummingbird vehicle, including the force and torque produced by each motor
- Program the test stand to regulate a desired pitch angle
- Compare control policy results simulation and hardware experiments.

2.2 Tasks

2.2.1 Measure quadrotor model parameters

The goal is to measure, or estimate, the unknown parameters in the quadrotor equations of motion that were derived in lecture. These unknown parameters are:

$$p = (m, J, l, k_F, k_M) \tag{2.1}$$

which represent the mass, moment of inertia matrix, length of the motor moment arm, motor thrust force coefficient, and motor aerodynamic moment coefficient, respectively.

2.2.2 Test stand simulation and control

The goal here is to simulate the one degree-of-freedom test stand. This test stand is created by allowing the hummingbird vehicle to only rotate about it's Y-axis

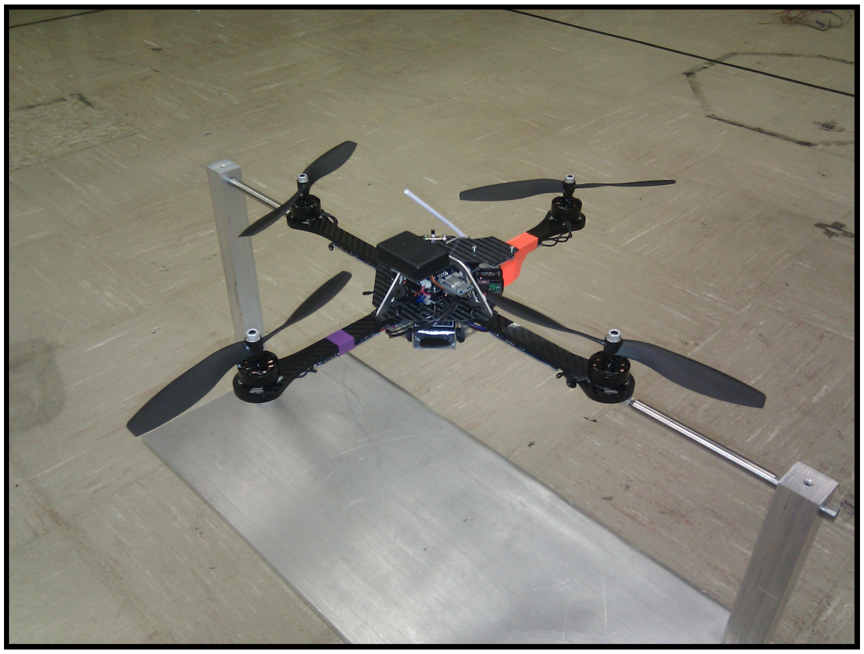

Figure 2.1: One degree-of-freedom (pitch) control test stand.

(pitch axis). After modeling the test stand, i.e. writing down the equations of motion that you derived in lecture and homework, you will also write a proportional-derivative (PD) controller to regulate the pitch angle of the vehicle.

2.2.3 Test stand onboard control

The goal here is to implement your controller that you developed in simulation on the real hardware. You will write your proportional-derivative controller in C, and test it's performance on the real vehicle.

2.3 Procedure

2.3.1 Measure quadrotor model parameters

1. Measure the mass of the hummingbird vehicle **without** the battery installed using the scale provided in the lab.

$$m_{vehicle} = \underline{\hspace{4cm}}$$

Measure the mass of the hummingbird battery

$$m_{battery} = \underline{\hspace{4cm}}$$

The total flight mass is

$$m = m_{vehicle} + m_{battery} = \underline{\hspace{3cm}}$$

2. Measure the distance from the motor rotation axis to the center of mass.

$$l = \underline{\hspace{3cm}}$$

3. The principle moments of inertia were extracted from a simplified CAD model.

$$J_{xx} = 4092.925 kg.mm^2$$
$$J_{yy} = 3944.437 kg.mm^2$$
$$J_{zz} = 7592.726 kg.mm^2$$

4. Measure the motor thrust coefficient k_F. At the workbench where the test stand is set up (Figure 2.2), run the hummingbird vehicle which will cycle the test motor through 15 steps of thrust, each lasting for 2 seconds. Run the Labview code which is provided in lab to capture and save motor rotations per second and thrust force (Figure 2.3). At the end of the cycle, hit the Stop button in Labview to save the captured data. Move the saved data file into the `N:/scratch/ae483/` folder, which you can then access in the same way from the other bench workstations. At your bench, be sure to move your data file to your `U:/` drive.

5. Measure the motor aerodynamic moment coefficient k_M. Use the second motor rig which measures the torque due to aerodynamic drag on the propeller. Just as before, at the workbench where the test stand is set up, run the hummingbird vehicle which will cycle the test motor through 15 steps of throttle, each lasting for 2 seconds. At the beginning of a cycle, run the Labview code which is provided in lab to capture and save motor rotations per second and thrust force. At the end of the cycle, hit the Stop button in Labview to save the captured data. Move the saved data file into the `N:/scratch/ae483/` folder, which you can then access in the same way from the other bench workstations. At your bench, be sure to move your data file to your `U:/` drive.

6. To compute k_F and k_M, plot the data from the LabView program in MATLAB as shown in Figure 2.4. Then find the average rotations per second for a given step. Multiply it by 2π and square the result. Then find the corresponding step in the force/moment data set and average the result. This is now one data point. Do this for 10 steps and plot the ten resulting points. You should notice that the points form a linear line. Use least squares regression to find the slope of this line. This slope is the k_F or k_M value. Show the TA your result.

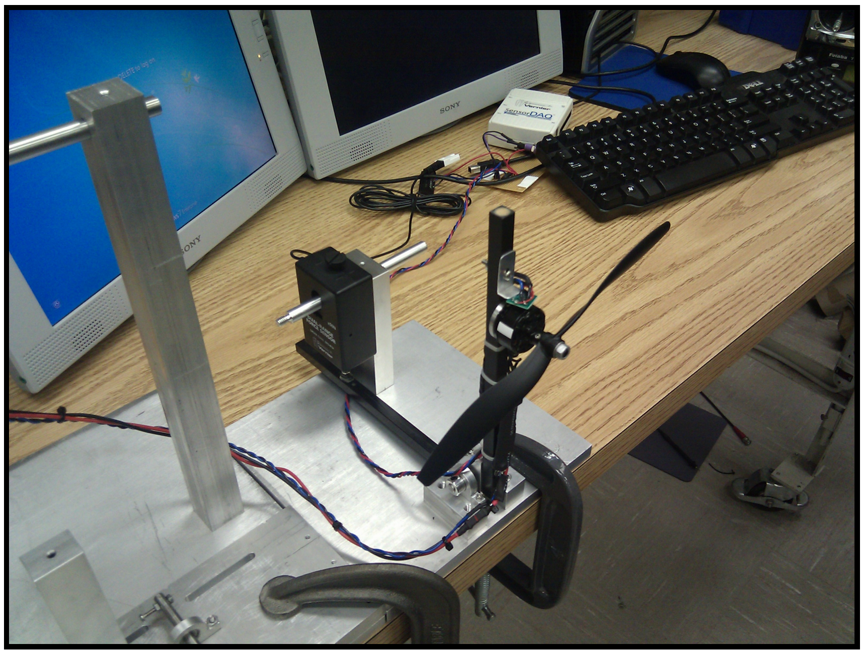

Figure 2.2: Motor thrust measurement rig.

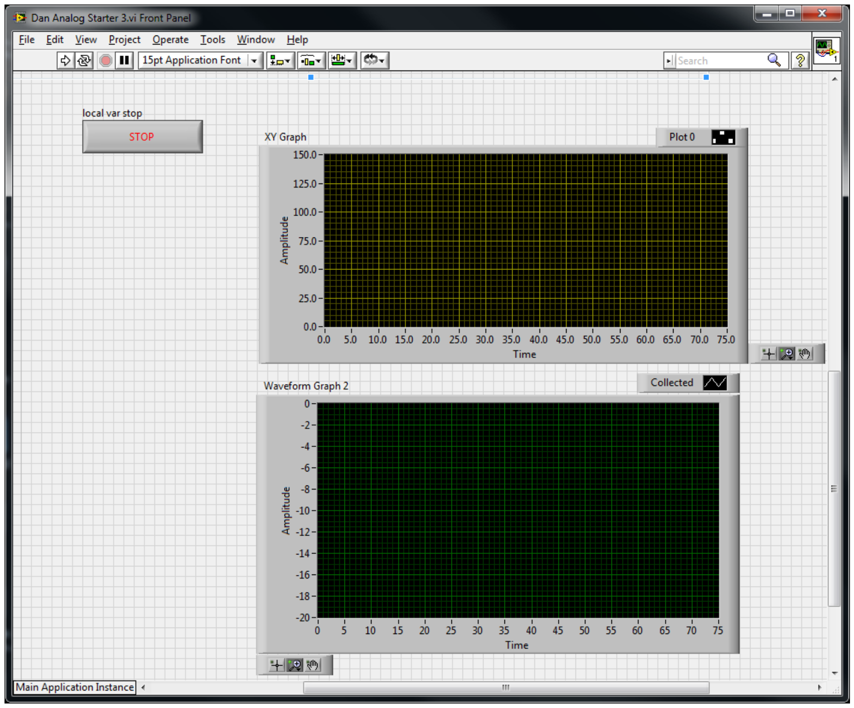

Figure 2.3: A Labview program provided in lab will allow you to capture and save motor rpm and motor thrust force.

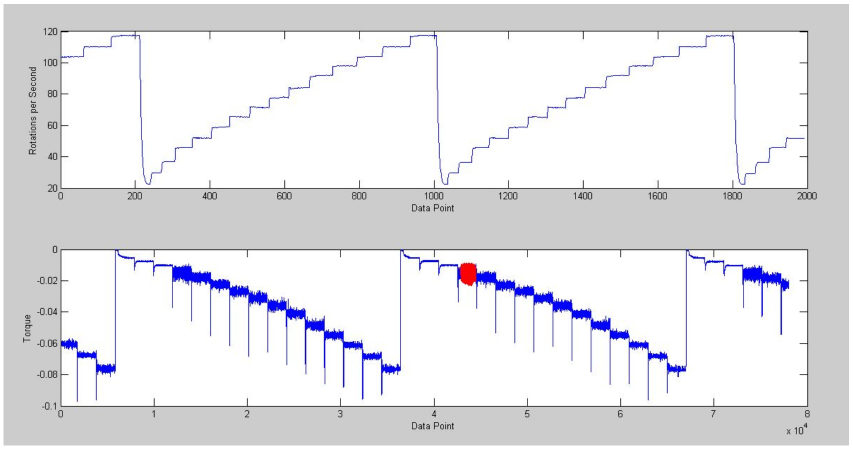

Figure 2.4: Plot of rotation per second data and torque LabView data.

2.3.2 Test stand simulation and control

1. Download `Lab2_codes.zip` from the course website. This zip archive contains a `Teststand` directory of MATLAB programs which provide the infrastructure of a simulator for the test stand. In particular, the script `simulate_this_teststand` will run the simulator. Your task is to modify the equations of motion function `Example_EOM_fun`, and the control policy `Example_policy_teststand`, which are both called from `simulate_this_teststand`.

2. Define the test stand equations of motion in `Example_EOM_fun.m`. The state and control vectors of the test stand are given by

$$x = \begin{bmatrix} \theta_2 \\ \dot{\theta}_2 \end{bmatrix} \qquad u = \begin{bmatrix} u_2 \end{bmatrix}$$

and the state-space form of the equations of motion are

$$\dot{x} = Ax + Bu$$

3. In `Example_policy_teststand.m` create a Proportional-Derivative (PD) controller that drives the test stand from arbitrary initial conditions to $(\theta_2, \dot{\theta}_2) = (0, 0)$. Assume a time horizon $t_f = 10$ seconds and initial condition

$$x(0) = \begin{bmatrix} \pi/4 \\ 0 \end{bmatrix}.$$

Recall the structure of a PD controller from lecture

$$u = -k_p\theta_2 - k_d\dot{\theta}_2$$

Iterate your choice of gains k_p and k_d until the resulting closed-loop trajectory achieves satisfactory performance (note the criteria you use to define satisfactory performace).

4. In `Example_policy_teststand.m` create a discrete-time Linear-Quadratic Regulator (LQR) that drives the test stand from arbitrary initial conditions to $(\theta_2, \dot{\theta}_2) = (0, 0)$. Assume initial condition

$$x(0) = \begin{bmatrix} \pi/4 \\ 0 \end{bmatrix}.$$

To create this controller, use the MATLAB `dlqr` command. Recall that the LQR controller has the form

$$u = -Kx$$

Note that the `dlqr` command has the following calling syntax

```
[K,S,e] = dlqr(A,B,Q,R)
```

where A and B are from the equations of motion, Q and R define the quadratic cost function, K is the optimal gain matrix, S is the infinite horizon solution of the associated discrete-time Riccati equation, and e are the closed-loop eigenvalues.

5. In `Example_policy_teststand.m` create a Proportional-Derivative (PD) controller that drives the test stand from arbitrary initial conditions to $(\theta_2, \dot{\theta}_2) = (a, 0)$, where a is any non-zero angle. Assume a time horizon $t_f = 10$ seconds and initial condition

$$x(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

6. In `Example_policy_teststand.m` create a discrete-time LQR controller that drives the test stand from arbitrary initial conditions to $(\theta_2, \dot{\theta}_2) = (a, 0)$, where a is any non-zero angle. Assume initial condition

$$x(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

2.3.3 Test stand hardware implementation

1. Proceed to the Test stand hardware to test your controller.
2. You will be using the Asctec Communication Interface (ACI) tool to test your controller. Note that this just implements

$$u = -k_p\theta_2 - k_d\dot{\theta}_2$$

where u here represents the applied pitch torque.

3. Tune your gains to achieve desirable performance for pitch control.
4. With the tuned gains, use the GroundStation code in Visual Studio 2010 to record messages data.

2.4 Checkoff Points

In lab, show the TA your:

1. computed model parameter vector.
2. onboard C code, `lab.c`, which regulates angle on the 1-DOF test stand.
3. code in action on the 1-DOF test stand.
4. MATLAB simulation equations of motion and control policy.

2.5 Report

Write a lab report which answers the following problems.

Problem 2.1. Using the test stand vehicle data collected in lab, compare hardware results versus simulation results. In particular, set the initial conditions and time span of the simulator so that they roughly match your experimental data. Plot θ_2 and $\dot{\theta}_2$ in two separate figures, where each figure contains plotted simulation and hardware trajectories.

Problem 2.2. Draw (in MATLAB) a 3D picture of the quad-rotor, given orientation from the onboard IMU, as viewed from the space frame. To complete this problem, you will be adding a small amount of code to the script `lab2_drawquad.m`, available on the course website. In particular, to animate the motion of the rigid body, you just need to update each vertex p of the rigid body. In the script, each column of the matrix `p1` is a vertex written in the coordinates of frame 1 (the body frame). Your goal is to generate the matrix `p0`, each column of which is the same vertex written in the coordinates of frame 0 (the world frame). Then, to display the result in MATLAB, you will generate the matrix `p2`, each column of which is the same vertex written in the coordinates of frame 2 (the MATLAB display frame). See Lab 1 for definitions of the coordinate frames.

Note: Your report should carefully describe anything that went "wrong" with the experiments, or any mismatch between the data and what you saw with your own eyes. It should be **written with your lab group**, and submitted by email to `ae483.fall16+`*`TA-name`*`@gmail.com` with the subject line "Lab 2 Report" and the attached report should be named with the last names of all group members "*Lastname1-Lastname2-Lastname3*-Lab-2.pdf". Copy all code into the report and submit just a single PDF file.