AE 483 Laboratory Note for Experiment (3)

FLIGHT CONTOL

by

Jacob Dray, Zach Fester, Steven Macenski

TA: Akshay Shetty

Lab AB6, Friday 3:00 - 4:50 p.m.

November 6, 2015

1. INTRODUCTION

The purpose of this experiment was to design a 6 degree-of-freedom flight controller capable of guiding a Hummingbird Quadrotor through two missions: stable hover and a return to stable hover after a disturbance or offset. To do this, a nested inner and outer control loop was implemented. First, the experiment was simulated with a series of MATLAB scripts and functions to tune the cost functions for the two loops for convergence within five seconds. Then, a C code was written with the same structure as the MATLAB loops to be flashed and readable by the Quadrotor. The outputted proportional and integral gains were then used and tested for this experimental system.

2. RESULTS AND DISCUSSIONS

3.1 Quad Rotor Flight Positions Data

Once flashed with the C code inner and outer loop controllers, the quad rotor achieved a stable hover condition about the origin of the experiment space. After a lateral or angular disturbance was applied, the rotor was able to return quickly with minimal oscillations back to stable hover at the origin. When first run, the first iteration of gains did not converge quickly enough putting the quad rotor in a state of constant oscillation. The cost functions dictating the proportional and derivative gains for our controller were changed accordingly so that the quadrotor converged with 2.5 seconds instead of 5 seconds. See Figure 1 for the flight path of the quadrotor during the run.

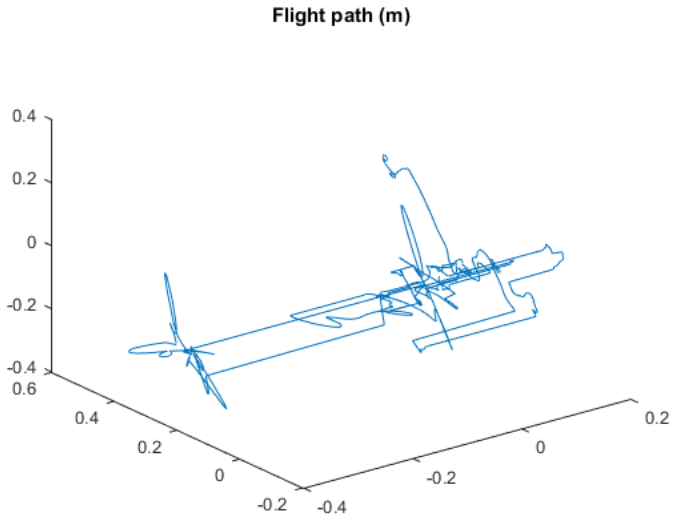


Figure 1: Flight Path

3.2 MATLAB Simulation with Position Offset

The control for the [4,4,4] positional offset is divergent. The controllers we built are effective for small angular disturbances and will converge quickly with our tuned gains. The controller, however, was not built for large position offsets that would require trajectory algorithms to discretize that path into sub-parts that the controller could deal with. The 4 meter offset in x,y,z directions is too large for the system to respond and converge. It can be seen with plots of much smaller offsets that convergence is achieved. Figures 2-7a show the divergence for a [4,4,4] offset while figures 2-7b show effective control for a 1 meter offset in each direction.

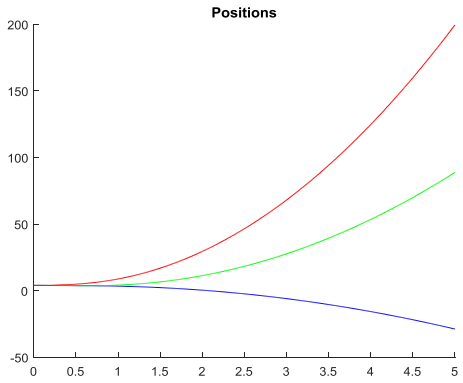


Figure 2a (left). Position [4,4,4] offset.

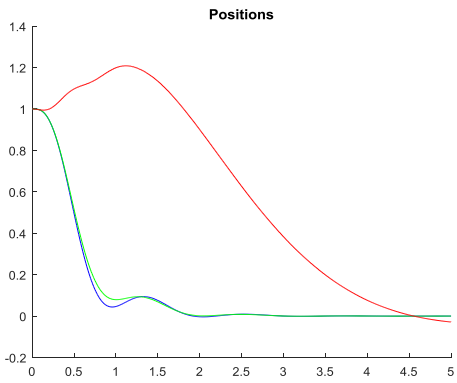


Figure 2b (right). Position [1,1,1] offset

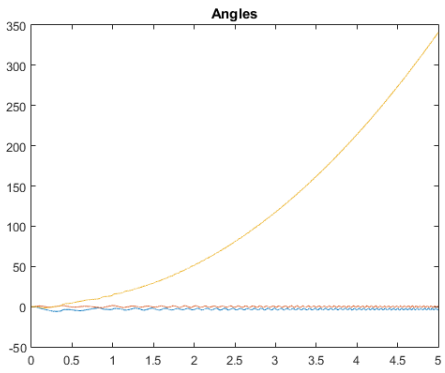


Figure 3a. Angles [4,4,4] offset.

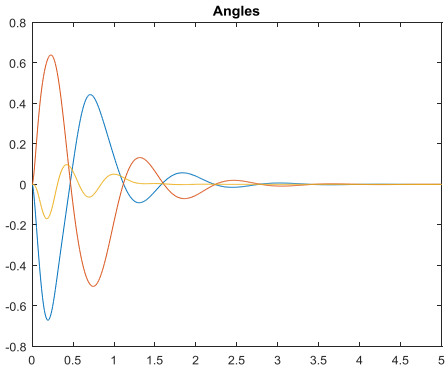


Figure 3b. Angles [1,1,1] offset

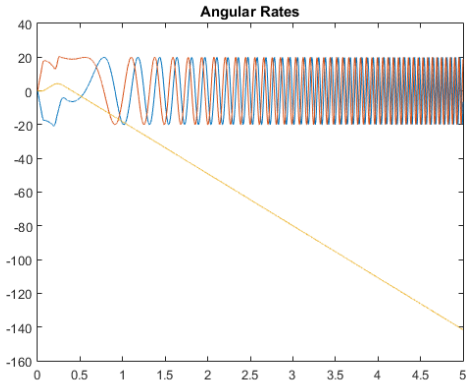


Figure 4a. Angular Rates [4,4,4] offset.

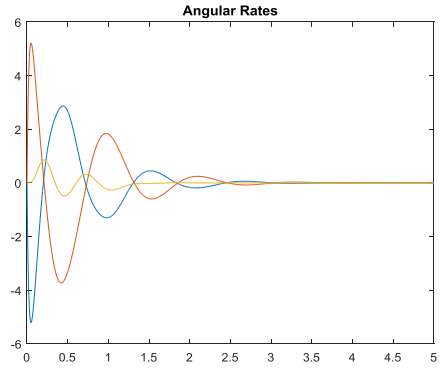


Figure 4b. Angular Rates [1,1,1] offset

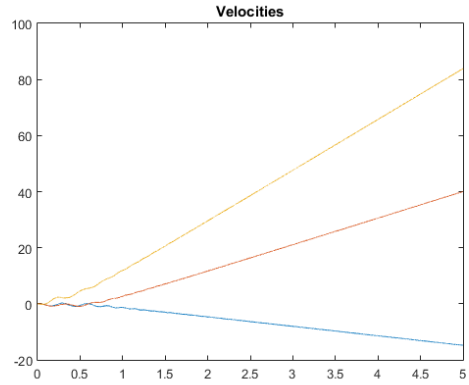


Figure 5a. Velocities [4,4,4] offset.

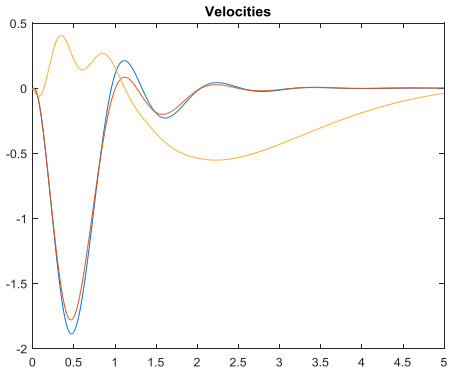


Figure 5b. Velocities [1,1,1] offset

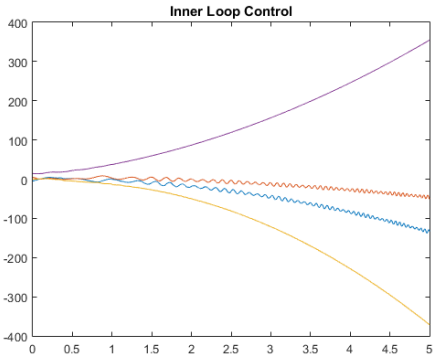


Figure 6a. Inner Loop Control [4,4,4] offset.

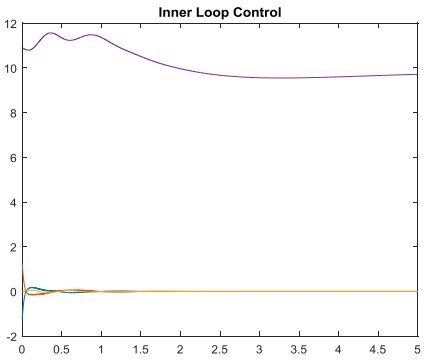
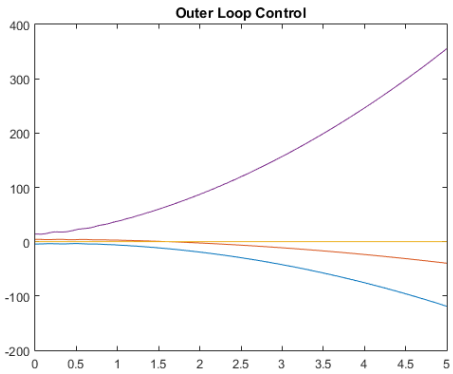


Figure 6b. Inner Loop Control [1,1,1] offset



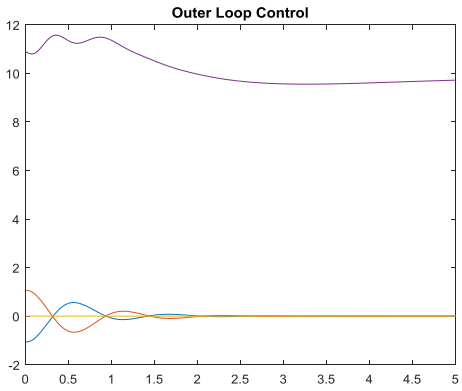


Figure 7a. Outer Loop Control [4,4,4] offset. **Figure 7b. Outer Loop Control [1,1,1] offset**

3.3 MATLAB Simulation of In-fight Test

```
clc
close all

load('mocap');
mocapn = transpose(mocap);

t = mocapn(1,:);
dt = mocapn(1,2:end) - mocapn(1,1:end-1);
x_pos = mocapn(6,:); y_pos = mocapn(7,:); z_pos = mocapn(8,:);
theta_x = mocapn(9,:); theta_y = mocapn(10,:); theta_z = mocapn(11,:);
q10 = mocapn(6:8,:);

%Use these mat files from Lab1
% CREATE WORLD OBJECTS
load('world.mat');  % this will load variables:   w0  wsz  wcolors

% CREATE A QUADROTOR
load('quadmodel.mat');  % this will load variables:  p1   faces   colors

R02 = [ 1  0  0; 0 -1  0; 0  0 -1 ];

% SETUP THE PLOT
clf;
set(gcf,'Renderer','zbuffer');
axis([-4 4 -4 4 -0.1 3.5]);
axis equal;
hold on;

R = [cos(theta_y(1))*cos(theta_z(1)), -cos(theta_y(1))*sin(theta_z(1)), sin(theta_y(1));
    cos(theta_x(1))*sin(theta_z(1))+cos(theta_z(1))*sin(theta_x(1))*sin(theta_y(1)), cos(theta_x(1))*cos(theta_z(1))-
sin(theta_x(1))*sin(theta_y(1))*sin(theta_z(1)), -cos(theta_y(1))*sin(theta_x(1));
    sin(theta_x(1))*sin(theta_z(1))-cos(theta_x(1))*cos(theta_z(1))*sin(theta_y(1)),
cos(theta_z(1))*sin(theta_x(1))+cos(theta_x(1))*sin(theta_y(1))*sin(theta_z(1)), cos(theta_x(1))*cos(theta_y(1))];
```

```
% ROTATE FROM BODY FRAME TO MATLAB PLOT FRAME
%%%%%
%Enter your code here:
%%%%%
p2 = R02*R*p1;

% ROTATE FROM WORLD FRAME TO MATLAB PLOT FRAME
w2 = R02*w0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
scatter3(w2(1,:), w2(2,:), w2(3,:), wsz, wcolors,'filled');
plot3(0,0,0,'k.','markersize',16);
h = patch('Vertices',p2','Faces',faces,...
        'CData',colors,'FaceColor','flat');
hTitle = title(sprintf('t = %4.2f',0));
lighting flat
light('Position',[0 -2 -1])
light('Position',[0 -2 1])
xlabel('x'); ylabel('y'); zlabel('z');
drawnow;
pause(0.5);

% ANIMATE THE RESULTS
i = 1;
tic;
while (i<length(t)-1)
   if (toc > dt(i))
      tic;
      i = i+1;
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % YOUR CODE HERE TO COMPUTE p0
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      R = [cos(theta_y(i))*cos(theta_z(i)), -cos(theta_y(i))*sin(theta_z(i)), sin(theta_y(i));
          cos(theta_x(i))*sin(theta_z(i))+cos(theta_z(i))*sin(theta_x(i))*sin(theta_y(i)), cos(theta_x(i))*cos(theta_z(i))-
sin(theta_x(i))*sin(theta_y(i))*sin(theta_z(i)), -cos(theta_y(i))*sin(theta_x(i));
          sin(theta_x(i))*sin(theta_z(i))-cos(theta_x(i))*cos(theta_z(i))*sin(theta_y(i)),
cos(theta_z(i))*sin(theta_x(i))+cos(theta_x(i))*sin(theta_y(i))*sin(theta_z(i)), cos(theta_x(i))*cos(theta_y(i))];

      % TRANSFORM FROM BODY TO WORLD FRAME
      %%%%%
      %Enter your code here:
      %%%%%
      p0 = R*p1;

      p0 = p0+repmat(q10(:,i),1,294);

      % TRANSFORM FROM WORLD FRAME TO MATLAB DISPLAY FRAME
      p2 = R02*p0;

      % UPDATE GRAPHICS OBJECT VERTICES
      set(h,'Vertices',p2');
      set(hTitle,'string',sprintf('t = %4.2f',t(i)));
      drawnow;
   end
end
```

3.4 Case Studies of Quad Rotor Performance

a. Quad_EOM contains the equations of motion of the quadcopter with relation to the state vector (position, velocity, angle, angular rate) being evaluated by the controllers. The Inner-Loop regulates attitude control with LQR controllers for roll, pitch, and yaw motion. The Outer-Loop receives the current state information via motion capture cameras. From there it gives Inner-Loop the desired input to achieve the desired state.

b. Initially, the quad-rotor receives position information from the motion-capture camera system. The quad-rotor is given a desired position, and then evaluates the difference between the desired and current position. This difference is then used as an input to derive the necessary torques which then generate the necessary thrust. This data is then relayed to the attitude control to tilt the rotor at the correct angle to begin a trajectory to achieve the desired position. Immediate convergence is not possible, therefore the quad-rotor will oscillate about the desired position at a rate based on the quality of the controller gains. Properly tuned gains will yield a smoother, quicker convergence.

c. If the mass defined in the outer-loop is less than the mass defined in the equations of motion, the quad-rotor will undercompensate for the perceived gravitational force and will descend too far. If the mass in the outer-loop is greater than the mass defined in the equations of motion, the quad-rotor Outer-Loop will ask the inner loop to apply too much thrust and it will rise too far.

d. For the position plot, the x and y lines will decrease from the initial positions to the desired positions, while z remains constant. The velocity plot will show negative velocity curves in x and y, due to the position of change of [5,1,1] to [0,0,1]. The magnitude of the x velocity will be greater than the y velocity due to the greater difference between initial/desired positions, while the z-velocity will remain at zero. The plot of the roll, pitch, and yaw angles will show an increase in roll (about x) and decrease in pitch (about y) from an initial zero position, then a respective decrease and increase back to zero where they will oscillate about the desired angle to maintain hover. There will be no change in yaw (about z). The angular velocity plots will mimic the behavior of the angle plots (positive spike for roll/ negative spike for pitch with zero yaw), but with small sharp spikes in angular velocity and faster convergence.

The plots of ux, uy, and uz show time-dependent behavior of the torque inputs. Ux would show an initial increase from zero followed by small oscillations about the desired input once the quad rotor is in the desired position at stable hover. Uy would show an initial decrease from zero followed by small oscillations about the desired input value eventually achieving stable hover. Uz would remain constant about zero to satisfy the zero yaw condition of the quad-rotor.

3. CONCLUSIONS

Overall, the method to split the control of the quad rotor into inner and outer loops yielded successful, safe flights. It is clear that using this method to decrease the order of the system analyzed is a viable option consistently used in industry applications. There is a limit to how far away the initial disturbance can be before trajectory algorithms must be applied. An improvement that could be made to this lab would be the expansion to a controller capable of handling larger disturbances.

4. REFERENCES

[1] “Flight Control Experiment” Laboratory Manual, AE 483, University of Illinois Urbana-Champaign, 2015