AE 483
Lab 2 10/9/2015
Jacob Dray, Zach Fester, Steven Macenski

Lab 2: Parameter Estimation and Introduction to Control

1. INTRODUCITON

The objective of this lab is the use the Hummingbird Quadrotor to estimate the parameters for the equations of motion. The quadrotor will be limited to one degree of freedom in an aluminum test stand, only allowing the quadrotor to pitch about the y-axis. A controller was designed to regulate this motion, simulated in MATLAB and then compared against experimental tests conducted in the lab with the quadrotor.

2. RESULTS AND ANALYSIS

2.1 Simulation vs. Experiment

The difference in performance between our simulation and experimental data can be attributed to the fact that in our code, we did not simulate the lag of acceleration of the propellers. Meaning that there was not an immediate application of input of force for the hardware. It took time for the rotors to speed up and slow down leading to overshoot of position and angular rate. Below, Figures 1 and 2 demonstrate the cases for both pitch and angular acceleration comparing our simulation and the actual results.

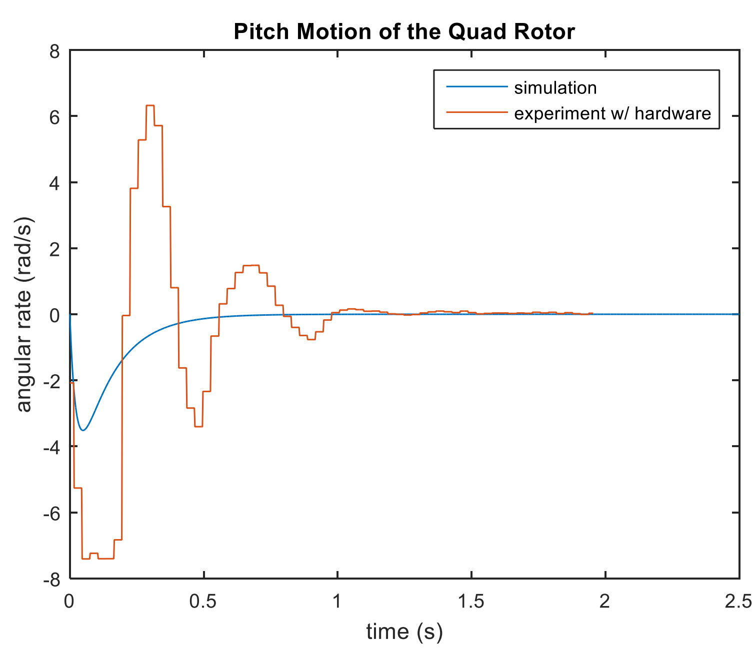

Figure 1. Pitch Position of the Quadrotor

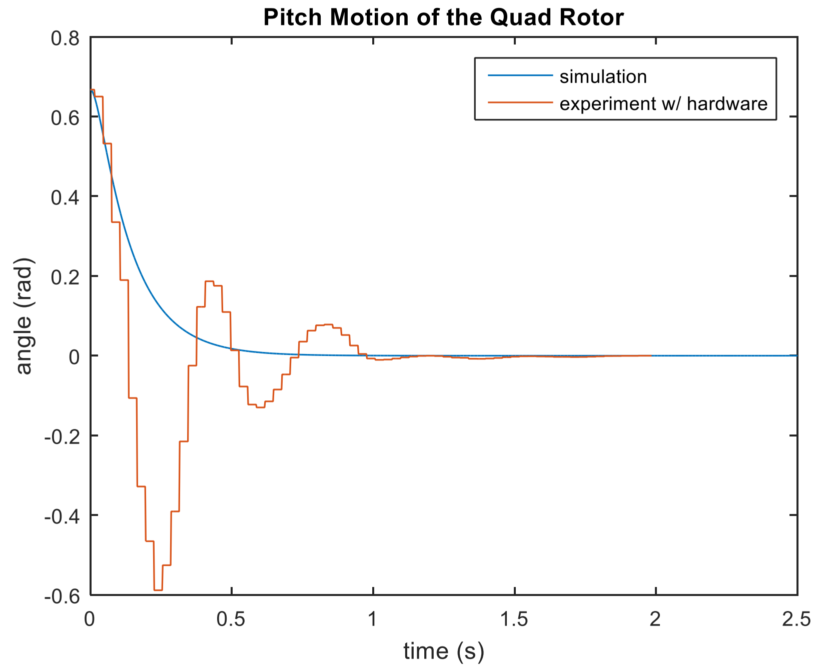

Figure 2. Angular Rate of the Quadrotor

2.2 Animation of Rigid Body Rotor

The animation of the rotor correctly displays the behavior observed in the lab section. After iterating our gains in simulation and then with the hardware, K_p = 1.30 and K_d = 0.20. It was observed that increasing K_p and decreasing K_d improved stability performance when a disturbance in pitch was applied.

APPENDIX A

Problem 1 Code

```
close all
clear all
home

%%% Initial conditions
tspan=[0 2.5]; dt=0.001;
x0 = [.5904+.077 0]';

%%% Identify your EOM function, policy function, and a force disturbance.
eom=@(t,x,u) Example_EOM_fun(x,u);
policy=@(t,x) Example_policy_teststandPart5(t,x);
%%% Default force disturbance is zeros.
disturbance=@(t,x) zeros(2,1);

%%%%% Discretize policy, run simulation. You shouldn't need to touch this
%%%%% section.
discrete_policy=@(t,x) discretize(policy,tspan,dt,t,x);
```

```
sys=@(t,x)vect4auto1(eom,discrete_policy,disturbance,t,x);
options=odeset('RelTol',1e-5,'MaxStep',dt);
tic; [T,X]=ode45(sys,tspan,x0,options); toc
[U_T,U]=discretize('Get Control History');
%%%%%


%%% Post-processing
plot_basic_teststand

%%% Clear persistent variables
% clear functions

 % Plot the Experiment vs The Simulation
 load('QuadTestStandData.mat');

time = DrayMacenskiFester(:,1);
theta = DrayMacenskiFester(:,2);
theta_dot = DrayMacenskiFester(:,5);

figure(1)
plot(T,X(:,1))
hold on
plotted_time = DrayMacenskiFester(1:992,1);
plotted_theta = DrayMacenskiFester(2423:3414,2)+.077;
plot(plotted_time,plotted_theta)
title('Pitch Motion of the Quad Rotor')
legend('simulation','experiment w/ hardware')
xlabel('time (s)')
ylabel('angle (rad)')
hold off


figure(2)
plot(T,X(:,2))
hold on
plotted_time2 = DrayMacenskiFester(1:992-15,1);

plotted_theta_dot = DrayMacenskiFester((2423+15):3414,5);
plot(plotted_time2,plotted_theta_dot)
hold off
title('Pitch Motion of the Quad Rotor')
legend('simulation','experiment w/ hardware')
xlabel('time (s)')
ylabel('angular rate (rad/s)')

figure(4)
plot(U_T,U)
title('Control')
```

Problem 2 Code

```
function [output] = lab2_drawquad()
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('QuadTestStandData.mat');
params.movie_filename = 'quad.avi';

max_t = 25.4132000000000;
t = DrayMacenskiFester(:,1);
theta = DrayMacenskiFester(:,2);
theta_dot = DrayMacenskiFester(:,5);
dt = 0:(DrayMacenskiFester(11380) - DrayMacenskiFester(11379)):max_t;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% CREATE A QUADROTOR
load('hw2code_quadmodel.mat');  % This will provide variables p1, faces,
colors


% SETUP THE PLOT
clf;
set(gcf,'Renderer','zbuffer');
axis([-1 1 -1 1 -0.5 1.5]);
axis equal;
hold on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% YOUR CODE HERE TO COMPUTE p2 TO INITIALIZE THE DRAWING

p0 = rotation(theta(1))*p1;
p2 = [1 0 0;...
      0 -1 0;...
      0 0 -1]*p0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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;
myV = VideoWriter(params.movie_filename);
myV.Quality = 100;
open(myV);
```

```
while (i<length(t)-1)
    %if (toc > dt(i))
        %tic;
        i = i+1;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % YOUR CODE HERE TO COMPUTE p2   it is easiest to first compute p0

        p0 = rotation(theta(i))*p1;

        p2 = [1 0 0;...
              0 -1 0;...
              0 0 -1]*p0;

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

         frame = getframe(gcf);
         writeVideo(myV,frame);

    %end
end
close(myV);
end %%% END lab2_drawquad()

function R = rotation(phi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% YOUR CODE HERE TO COMPUTE ROTATION MATRIX R_1^0 FROM ZYX Euler Angles

R = [cos(phi) 0 sin(phi);...
     0 1 0;...
     -sin(phi) 0 cos(phi)];

end
```