Aero 2 Half-Quadrotor PD Control Design

In this lab we will design a PD controller according to a set of specifications and simulate the closed-loop response to ensure it matches those specifications. We will then run the PD controller on the Aero 2 system using QUARC.

Concept Review

PD Control Design

We can the Proportional-Derivative (PD) control shown in Figure 1 to stabilize the yaw position to a setpoin:

$$u = k_p(\psi_d(t) - \psi(t)) - k_d\dot{\psi}(t),$$

where $\psi(t)$ is the yaw angle, $\psi_d(t)$ is the yaw reference command, and $u(t)$ is the voltage applied to the both rotors.

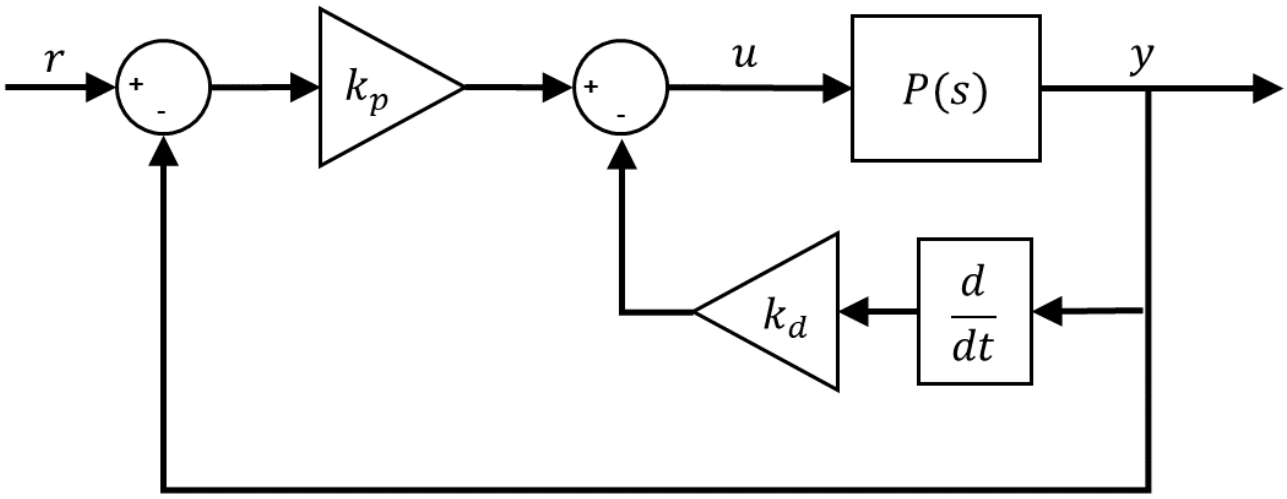


Figure 1: PD Control

The Laplace transform of the PD control given above yields

$$U(s) = k_p(\Psi_d(s) - \Psi(s)) - k_d s\Psi(s).$$

Recall from the *Parameter Estimation* lab that the transfer function model of the half-quadrotor system is

$$\frac{\Psi(s)}{U(s)} = \frac{D_t K_f}{J_y s^2 + D_y s}.$$

Substituting the control into the plant transfer function model we get

$$Y(s) = \frac{D_t K_f}{J_y s^2 + D_y s}(k_p(R(s) - Y(s)) - k_d s Y(s)).$$

Solving for $Y(s)/R(s)$, we obtain the closed-loop expression

$$\frac{D_t K_f k_p}{J_y s^2 + (D_y + D_t K_f k_d)s + D_t K_f k_p}.$$

Given that this is a second-order transfer function, we can find its natural frequency and damping ratio parameters based on prototype second-order function

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

We can design the PD control gains according to a specified natural frequency, ω_n, and damping ratio, ζ, with the equations

$$k_p = \frac{J_y \omega_n^2}{D_t K_f}$$

and

$$k_d = -\frac{D_y - 2J_y\zeta\omega_n}{D_t K_f}.$$

Desired closed-loop response specifications

1. Peak time: $t_p \leq 3.5$ s.
2. Percent Overshoot: $PO \leq 2.5\%$.

```
% Peak time and overshoot specifications
PO = 2.5;
tp = 3.5;
```

Peak Time and Overshoot Equations

We can use the following expressions to obtain the required ω_n and ζ from the peak time and overshoot specifications. In a second-order system, the amount of overshoot depends solely on the damping ratio parameter

$$PO = 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$$

The peak time depends on both the damping ratio and natural frequency of the system and it can be derived as:

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$$

```
% Damping ratio from overshoot specification.
zeta = -log(PO/100) * sqrt( 1 / ( ( log(PO/100) )^2 + pi^2 ) );
% Natural frequency from specifications (rad/s)
wn = pi / ( tp * sqrt(1-zeta^2) );
```

Find the PD Gains

Given the peak time and percent overshoot response specifications, use the overshoot and peak time equations to find the necessary damping ratio and natural frequency and then calculate the PD gains accordingly.

Load the Aero 2 Parameters

```
aero2_parameters;
```

Load damping and thrust parameters values.

```
aero2_half_quad_id_param;
```

Calculate the gains

```
% PD Design for Half-Quadrotor yaw control
kp = Jy*wn^2/(Kf*Dt)
```

```
kp = 106.8829
```

```
kd = -Jy*(Dy/Jy-2*wn*zeta)/(Dt*Kf)
```

```
kd = 111.5379
```

PD Closed-Loop Simulation

The `s_aero2_half_quad_pd.slx` Simulink model shown in Figure 2 uses the *PIV Controller* block from the *QUARC Targets* library to implement the half-quadrotor PD control. The *Aero 2 Half-Quadrotor Model* subsystem includes a block diagram model of the system.

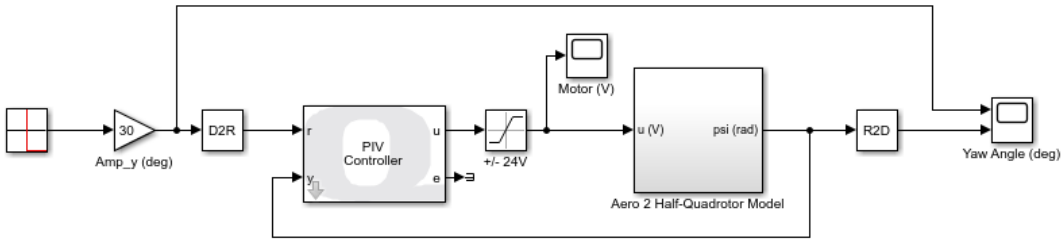


Figure 2: The `s_aero2_half_quad_pd.slx` Simulink model simulated the Aero 2 PD control response.

Simulate the closed-loop response in Simulink.

```
open("s_aero2_half_quad_pd.slx")
sim("s_aero2_half_quad_pd.slx")
```

See the sample response shown in Figure 3.

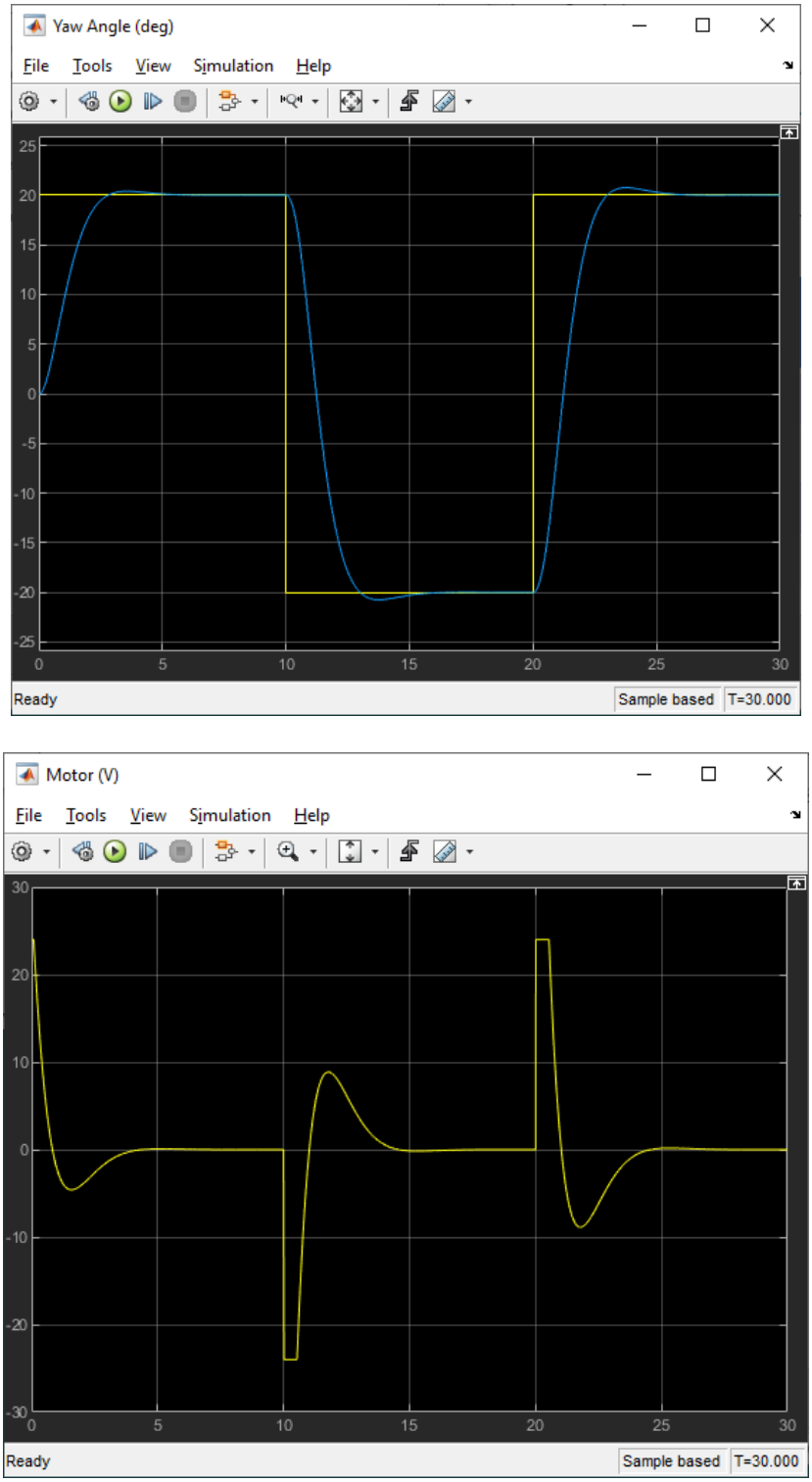
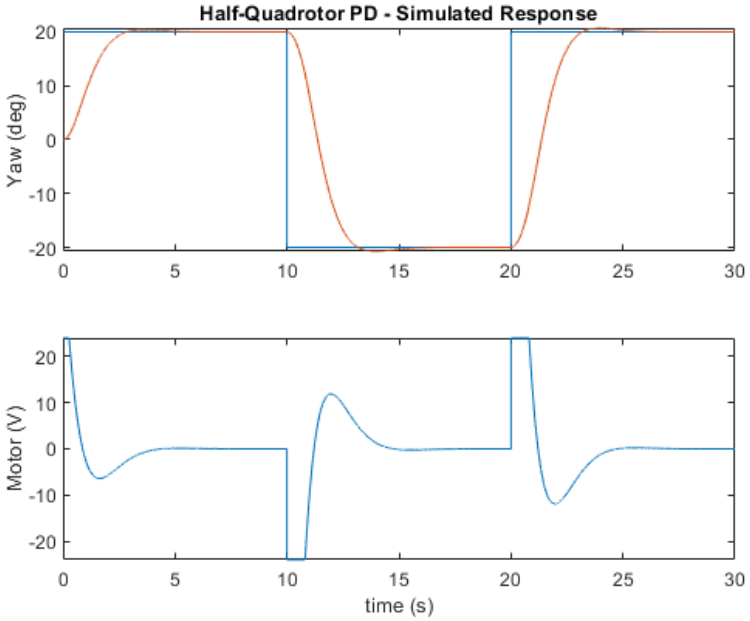


Figure 3: Aero 2 PD control closed-loop response simulation.

Plot results from the logged data.

```
% Load measured data from Simulink scopes
t_sim = DataHalfQuadPDSimVm.time; % time (s)
u_sim = DataHalfQuadPDSimVm.signals.values; % pitch/front motor voltage (V)
psi_d_sim = DataHalfQuadPDSimYaw.signals(1).values; % desired yaw angle (rad)
```

```
psi_sim = DataHalfQuadPDSimYaw.signals(2).values; % yaw angle (rad)
% plot
subplot(2,1,1);
plot(t_sim,psi_d_sim,t_sim,psi_sim);
title('Half-Quadrotor PD - Simulated Response');
ylabel('Yaw (deg)');
subplot(2,1,2);
plot(t_sim,u_sim);
ylabel('Motor (V)');
xlabel('time (s)');
```



The peak time and percent overshoot of the system are:

$t_p = 23.88 - 20 = 3.89$ s

$$PO = 100 \times \frac{20.71 - 20}{40} = 1.77\%$$

The percent overshoot control design specifications is satisfied in simulation, but the peak time is not, i.e., > 3.5 s). This is due to the actuator saturation that occurs at this setpoint. Using a setpoint of 10 deg results in compliant results. Given this, the control design is satisfied and the hardware test can be performed.

PD Control Hardware Implementation

Run the PD controller on the Aero 2 half-quadrotor system using the `q_aero2_half_quad_pd.slx` Simulink model, shown in Figure 4, with QUARC.

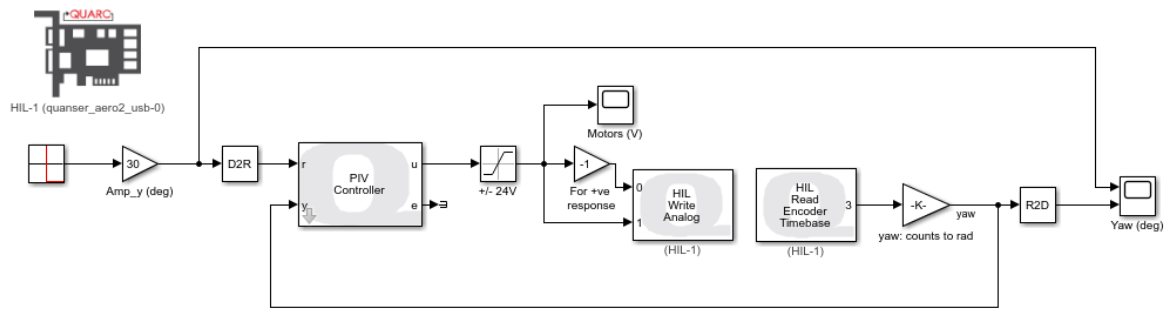


Figure 4: The `q_aero2_half_quad_pd.slx` Simulink model uses QUARC to run the PD control on the Aero 2

Aero 2 Setup

1. Make sure the Aero 2 has been tested as instructed in the Quick Start Guide.
2. Launch MATLAB and browse to the working directory that includes the Simulink models for this lab.
3. Configure the Aero 2 in the Half-Quadrotor configuration:
4. **Lock** the pitch axis and **unlock** the yaw axis.
5. Both rotors are **horizontal** (i.e., rotor shields are parallel with the ground).
6. Mount weight on each rotor.
7. Connect the USB cable to your PC/laptop.
8. Connect the power and turn the power switch ON. The Aero base LED should be red

Build and run the following Simulink model in QUARC by clicking on the *Monitor & Tune* button.

```
open("q_aero2_half_quad_pd.slx");
```

See the sample response in Figure 5.

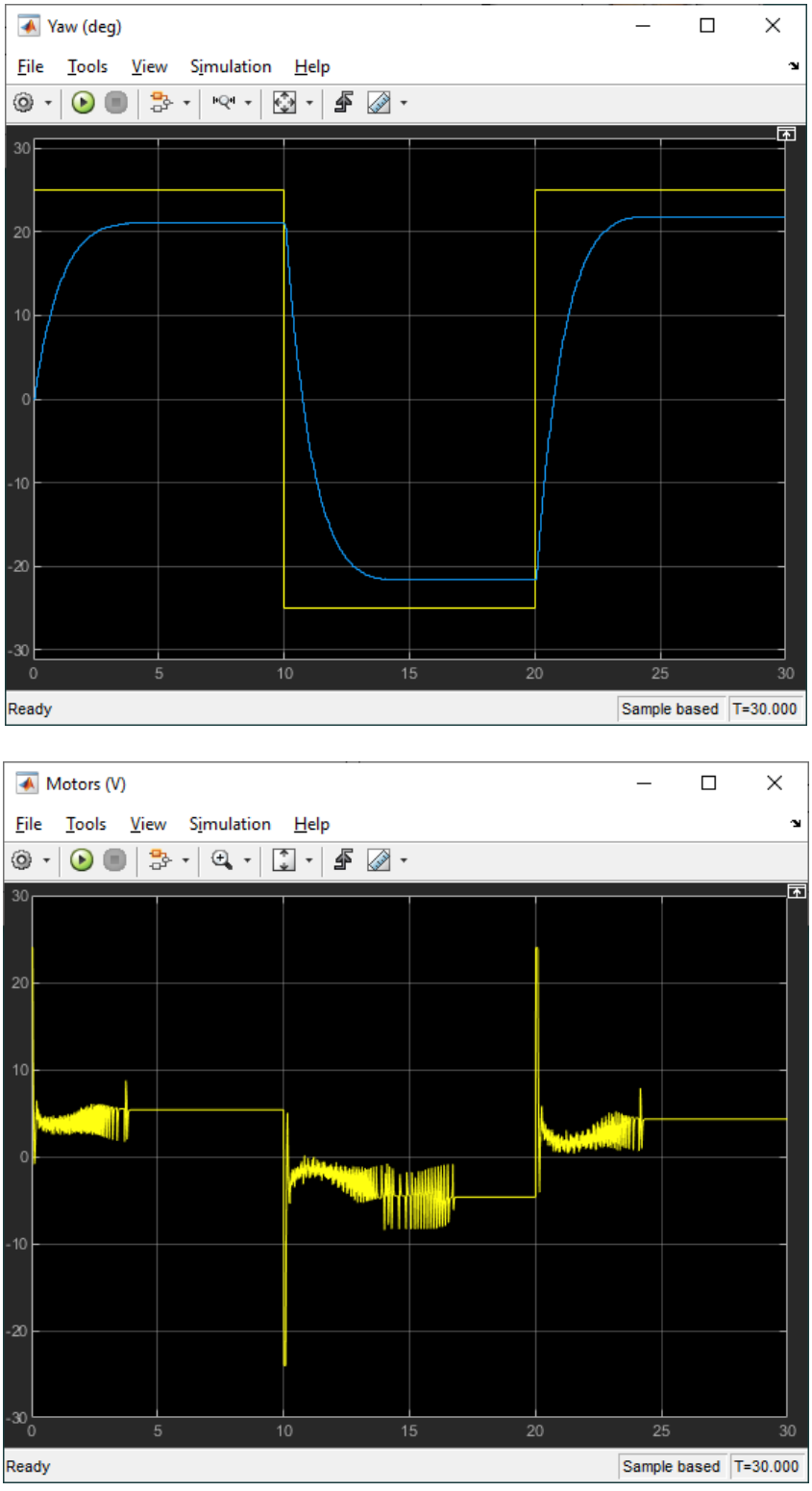
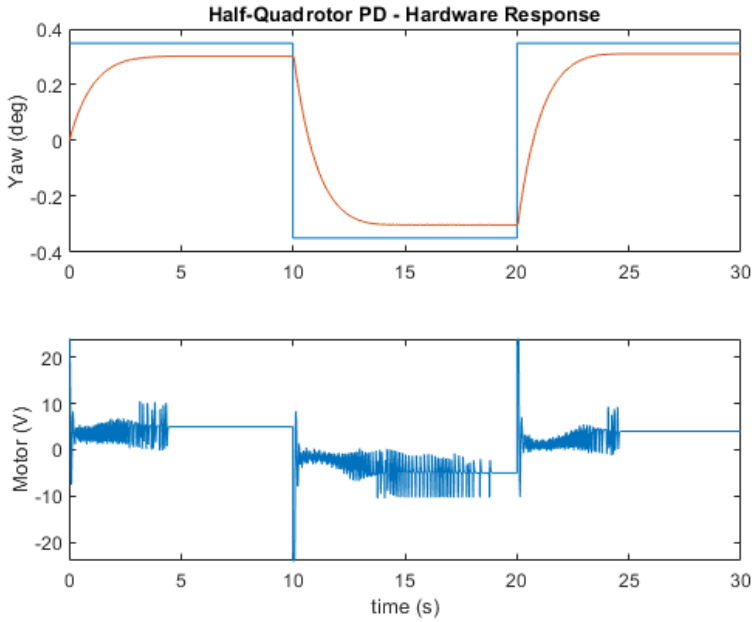


Figure 5: Sample PD control response on Aero 2.

Plot results from saved .MAT file.

```
% Load measured data from past run
load('DataHalfQuadPD.mat');
% store in variables
```

```
t = DataHalfQuadPD(1,:); % time (s)
u = DataHalfQuadPD(2,:); % pitch/front motor voltage (V)
psi_d = DataHalfQuadPD(3,:); % desired yaw angle (rad)
psi = DataHalfQuadPD(4,:); % yaw angle (rad)
%
subplot(2,1,1);
plot(t,psi_d,t,psi);
title('Half-Quadrotor PD - Hardware Response');
ylabel('Yaw (deg)');
subplot(2,1,2);
plot(t,u);
ylabel('Motor (V)');
xlabel('time (s)');
```



The peak time and percent overshoot of the system are:

$t_p = 24.30 - 20 = 4.30$ s

$PO = 0\%$

The control design specifications are satisfied on the hardware using the PD control. Compared to the simulated response, the response on the hardware has a steady-state error and less overshoot. Both of these observations are probably due to the unmodeled friction (e.g. Coulomb friction) about the yaw axis. Increasing the proportional gain, introducing an integrator, or using a more advanced friction-compensation scheme could minimize this.