Aero 2: 2 DOF PD Control

Concept Review

In this lab, a proportional-derivative (PD) control is used to control the pitch and yaw axes to a desired angle.

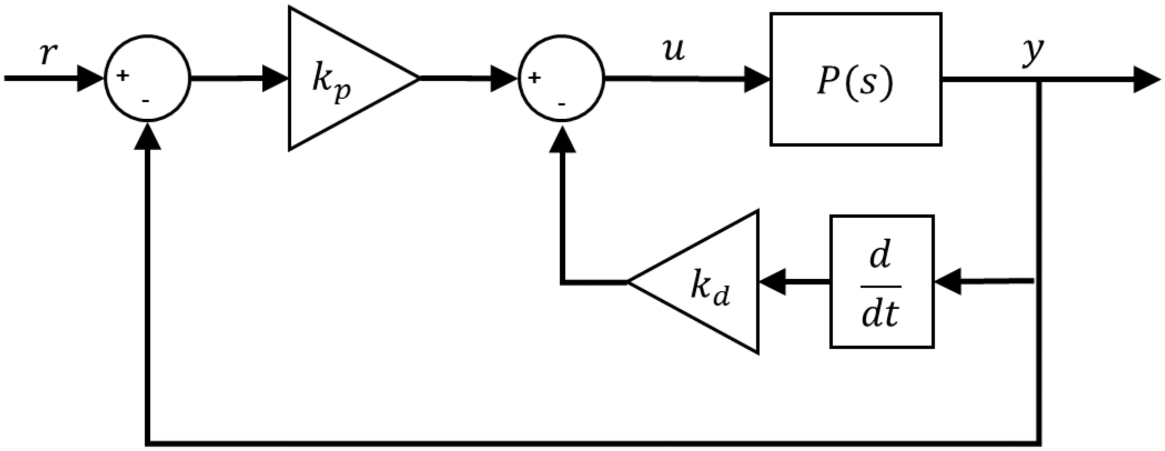


Figure 1: Proportional-Derivative (PD) control

This is a variation of the standard PD control where only the negative velocity is fed back, i.e., not the velocity of the *error*, and a low-pass filter will be used in-line with the derivative term to suppress measurement noise. The PD control shown in Figure 1 has the following structure

$$u = k_p(r(t) - y(t)) - k_d\dot{y}(t), \qquad (1)$$

where k_p is the proportional gain, k_d is the derivative (velocity) gain, $r = \theta_d(t)$ is the reference pitch angle, $y = \theta(t)$ is the measured pitch angle, and $u = V_p(t)$ is the control input - the applied motor voltage to the front pitch rotor.

For the PD control of the yaw axis, the reference is the desired yaw angle $r = \psi_d(t)$, the measured variable is the yaw angle $y = \psi(t)$, and the control input is the rear yaw rotor voltage $u = V_y(t)$.

Pitch and Yaw Models

The pitch and yaw can be modeled as separate transfer functions:

$$P_p(s) = \frac{\Theta(s)}{V_p(s)} = \frac{\frac{D_t K_{pp}}{J_p}}{s^2 + \frac{D_p}{J_p}s + \frac{K_{sp}}{J_p}}$$

and

$$P_y(s) = \frac{\Psi(s)}{V_y(s)} = \frac{\frac{D_t K_{yy}}{J_y}}{s^2 + \frac{D_y}{J_y} s}$$

Control Design

The PD controller transfer function can be found by taking the Laplace transform of Equation 1. Since the Aero 2 starts at rest, the initial conditions are zero, i.e. $\theta(0^-) = 0$ and $\dot{\theta}(0^-) = 0$, and we can obtain the following

$$U(s) = k_p(R(s) - Y(s)) - k_d s Y(s).$$

Given the control input is the front/pitch motor voltage, $U(s) = V_p(s)$. The closed-loop transfer function can be found by substituting the PD control into the pitch transfer function, and solving for $Y(s)/R(s)$:

$$\frac{Y(s)}{R(s)} = \frac{\frac{D_t K_{pp} k_p}{J_p}}{s^2 + \frac{D_p + D_t K_{pp} k_d}{J_p} s + \frac{K_{sp} + D_t K_{pp} k_p}{J_p}}. \qquad (2)$$

Given the standard second-order prototype transfer function

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad (3)$$

we can express the PD control gains based on the required natural frequency, ω_n, and damping ratio, ζ, with the equations

$$k_p = \frac{J_p \omega_n^2 - K_{sp}}{D_t K_{pp}} \qquad (4)$$

and

$$k_d = \frac{2 J_p \omega_n \zeta - D_p}{D_t K_{pp}}. \qquad (5)$$

We can apply the same control design for the yaw-axis and get the PD equations:

$$k_p = \frac{J_y \omega_n^2 K_{sp}}{D_t K_{yy}} \qquad (6)$$

and

$$k_d = \frac{2 J_y \omega_n \zeta - D_y}{D_t K_{yy}}. \qquad (7)$$

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

$$PO = 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \qquad (8)$$

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

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \qquad (9)$$

Generally speaking, the damping ratio affects the shape of the response while the natural frequency affects the speed of the response.

Decoupled control design: Note that this does not take into account the reaction torque effects as documented in the *2 DOF Helicopter Modeling* lab. The PD control is designed separately for the pitch and yaw axes, i.e. each axes is treated as a single-input, single-output (SISO) system. This is known as a *decoupled controller*. As a result, simultaneously controlling the pitch and yaw axes to a desired reference command may yield unexpected motions. For example, large overshoot can be seen in the pitch axis as the yaw is tracking a reference command signal.

PD Control Design

Control Specifications

Design PD gains according to the following set of specifications:

1. Peak time: $t_p \leq 2.5$ s for pitch angle and $t_p \leq 3.5$ s for yaw angle.
2. Percent Overshoot: $PO \leq 5\%$ for both axes.

Find the second-order time-domain design requirements based on the desired peak time and overshoot specifications.

Pitch Axis

```
% Peak time and overshoot specifications
PO_p = 5;
tp_p = 2.5;
% Damping ratio from overshoot specification.
zeta_p = -log(PO_p/100) * sqrt( 1 / ( ( log(PO_p/100) )^2 + pi^2 ) );
% Natural frequency from specifications (rad/s)
wn_p = pi / ( tp_p * sqrt(1-zeta_p^2) );
```

Yaw Axis

```
% Peak time and overshoot specifications
PO_y = 5;
tp_y = 3.5;
% Damping ratio from overshoot specification.
```

```
zeta_y = -log(PO_y/100) * sqrt( 1 / ( ( log(PO_y/100) )^2 + pi^2 ) );
% Natural frequency from specifications (rad/s)
wn_y = pi / ( tp_y * sqrt(1-zeta_y^2) );
```

Find PD Gains

Load the AERO 2 Parameters

```
aero2_parameters;
```

Load stiffness, damping, and thrust parameters values.

```
aero2_parameters_id;
```

Find the PD gains for pitch axis based on equations 4 and 5

```
% Pitch proportional gain (V/rad)
kp_p = (Jp*wn_p^2-Ksp)/Kpp/Dt
```

```
kp_p = 116.2420
```

```
% Pitch Derivative gain (V-s/rad)
kd_p = (2*zeta_p*wn_p*Jp-Dp)/Kpp/Dt
```

```
kd_p = 99.6990
```

Find the PD gains for the yaw using equations 6 and 7.

```
% Yaw proportional gain (V/rad)
kp_y = Jy*wn_y^2/Kyy/Dt
```

```
kp_y = 35.8626
```

```
% Yaw derivative gain (V-s/rad)
kd_y = (2*zeta_y*wn_y*Jy-Dy)/Kyy/Dt
```

```
kd_y = 38.0289
```

PD Control Simulation

The `s_aero2_2dof_pd` Simulink model shown in Figure 2 can be used to simulate the pitch and yaw PD control of the Aero 2 - 2 DOF Helicopter. The Simulink model uses the *PIV Controller* block from the *QUARC Targets* library to implement the PD control designed and includes the motor limits of ± 24V to represent the system more accurately.

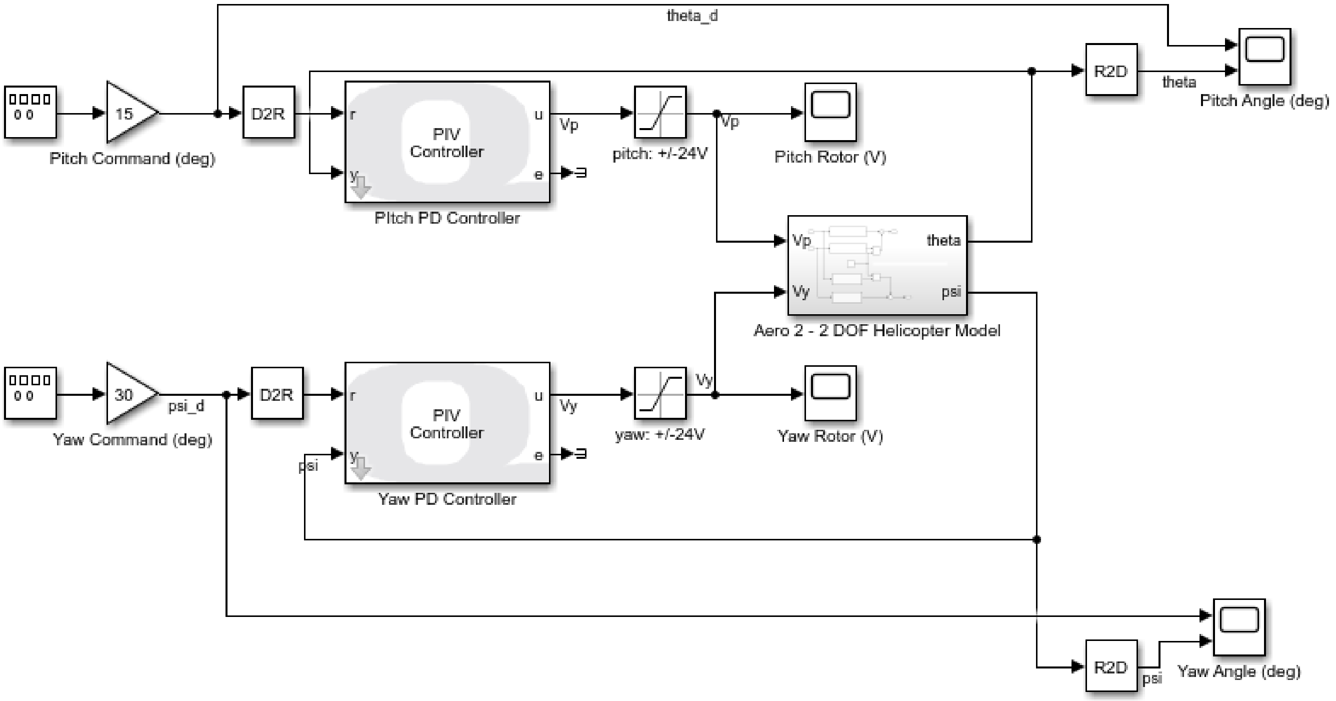


Figure 2: Simulates the PD control of the pitch and yaw axes on the 2 DOF Aero 2.

The plant is modeled in the *Aero 2 - 2 DOF Helicopter Model* subsystem using the transfer functions documented in *Aero2_2DOF_Modeling,* as shown in Figure 3. Use the *Manual Switch* blocks to apply or remove the cross-axis coupling effect from the reaction torques.

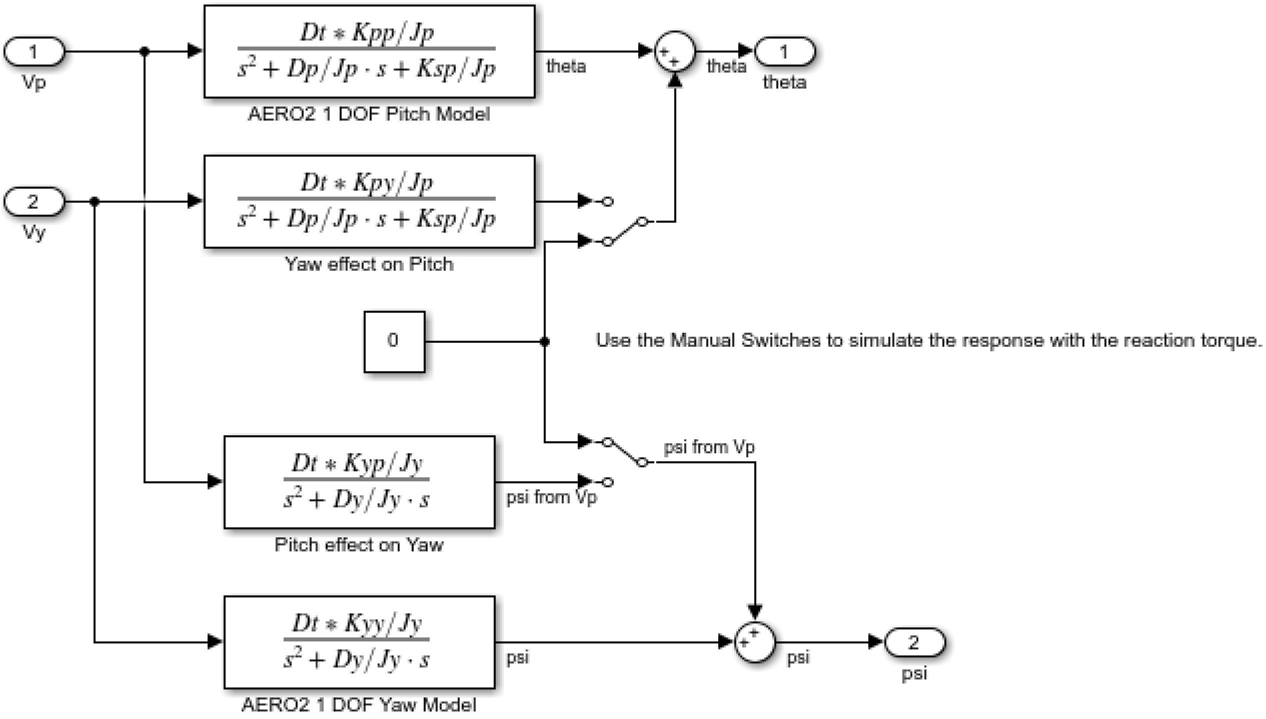

1
Vp
2
Vy
$\frac{Dt*Kpp/Jp}{s^2+Dp/Jp\cdot s+Ksp/Jp}$
AERO2 1 DOF Pitch Model
theta
theta
1
theta
$\frac{Dt*Kpy/Jp}{s^2+Dp/Jp\cdot s+Ksp/Jp}$
Yaw effect on Pitch
0
Use the Manual Switches to simulate the response with the reaction torque.
psi from Vp
$\frac{Dt*Kyp/Jy}{s^2+Dy/Jy\cdot s}$
psi from Vp
Pitch effect on Yaw
$\frac{Dt*Kyy/Jy}{s^2+Dy/Jy\cdot s}$
psi
AERO2 1 DOF Yaw Model
psi
2
psi

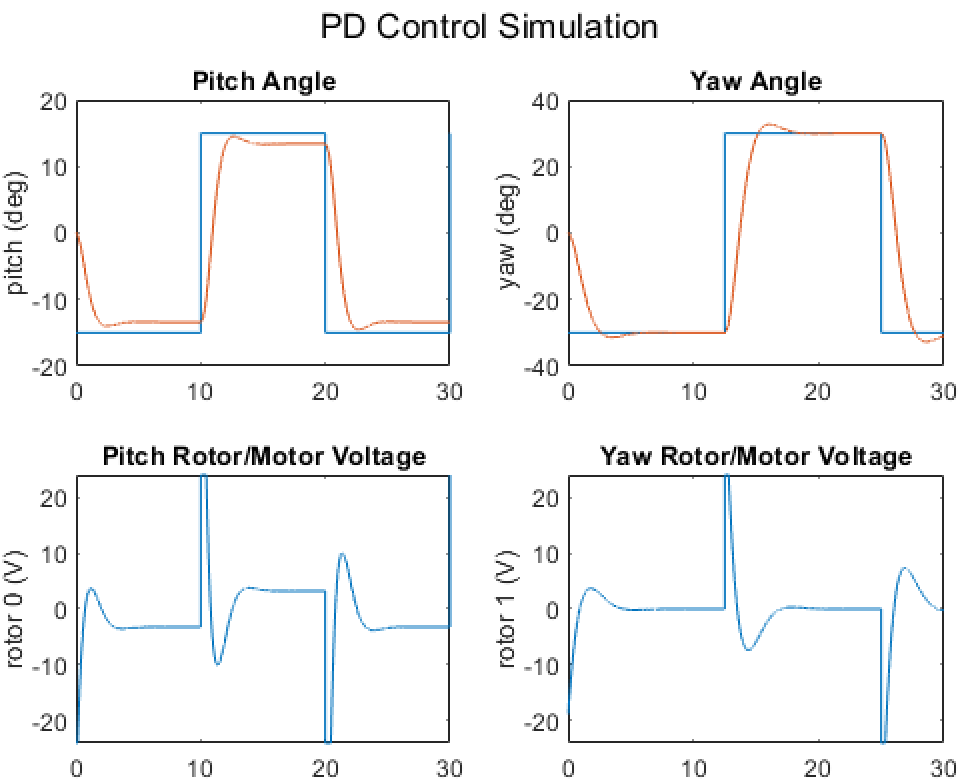
Figure 3: Aero 2 - 2 DOF Helicopter model used for the simulation

Run the Simulnk simulation and verify that the simulated response satisfies the design specifications.

```
% Simulated closed-loop PID control using full model w/ actuator limits
open("s_aero2_2dof_pd.slx");
sim("s_aero2_2dof_pd.slx");
```

Plot response using the data logged from the Simulink scopes.

```
t = Aero2_PD_Sim_Vp.time; % time (s)
Vp = Aero2_PD_Sim_Vp.signals.values; % pitch voltage (V)
theta_d_sim = Aero2_PD_Sim_Pitch.signals(1).values; % desired/command pitch angle (rad)
theta_sim = Aero2_PD_Sim_Pitch.signals(2).values; % pitch angle response (rad)
Vy = Aero2_PD_Sim_Vy.signals.values; % yaw voltage (V)
psi_d_sim = Aero2_PD_Sim_Yaw.signals(1).values; % desired/command yaw angle (rad)
psi_sim = Aero2_PD_Sim_Yaw.signals(2).values; % yaw angle response (rad)
%
subplot(2,2,1);
plot(t,theta_d_sim,t,theta_sim);
title('Pitch Angle');
ylabel('pitch (deg)');
subplot(2,2,2);
plot(t,psi_d_sim,t,psi_sim);
title('Yaw Angle');
ylabel('yaw (deg)');
subplot(2,2,3);
plot(t,Vp);
title('Pitch Rotor/Motor Voltage');
ylabel('rotor 0 (V)');
subplot(2,2,4);
plot(t,Vy);
title('Yaw Rotor/Motor Voltage');
ylabel('rotor 1 (V)');
sgtitle('PD Control Simulation');
```



Simulated Response Results

The peak time and percentage overshoot for the pitch axis are:

$$t_p = 2.62 \text{ s and } PO = \frac{14.52 - 13.4}{13.4 - (-13.4)} \times 100 = 4.18\%.$$

Based the yaw step response starting at 12.5 s, the peak time and percentage overshoot are

$$t_p = 3.60 \text{ s and } PO = \frac{32.7 - 30}{60} \times 100 = 6.17\%.$$

The peak time for both the pitch and the yaw are slightly slower than our specifications. This is most likely due to motor saturation, i.e., the control input exceeds the ± 24 motor limits. The percent overshoot of the pitch is within the limits but the yaw exceeds the maximum overshoot slightly.

Since the specifications are nearly satisfied and that we expect the response on the hardware to be different, i.e., since the model is an approximation and does not account for all the Aero 2 dynamics, the control can be tested on the Aero 2 hardware.

Running the PD Control on the Hardware

The `q_aero2_2dof_pd` Simulink model shown in Figure 4 will be used with QUARC to run the PD control on the Aero 2.

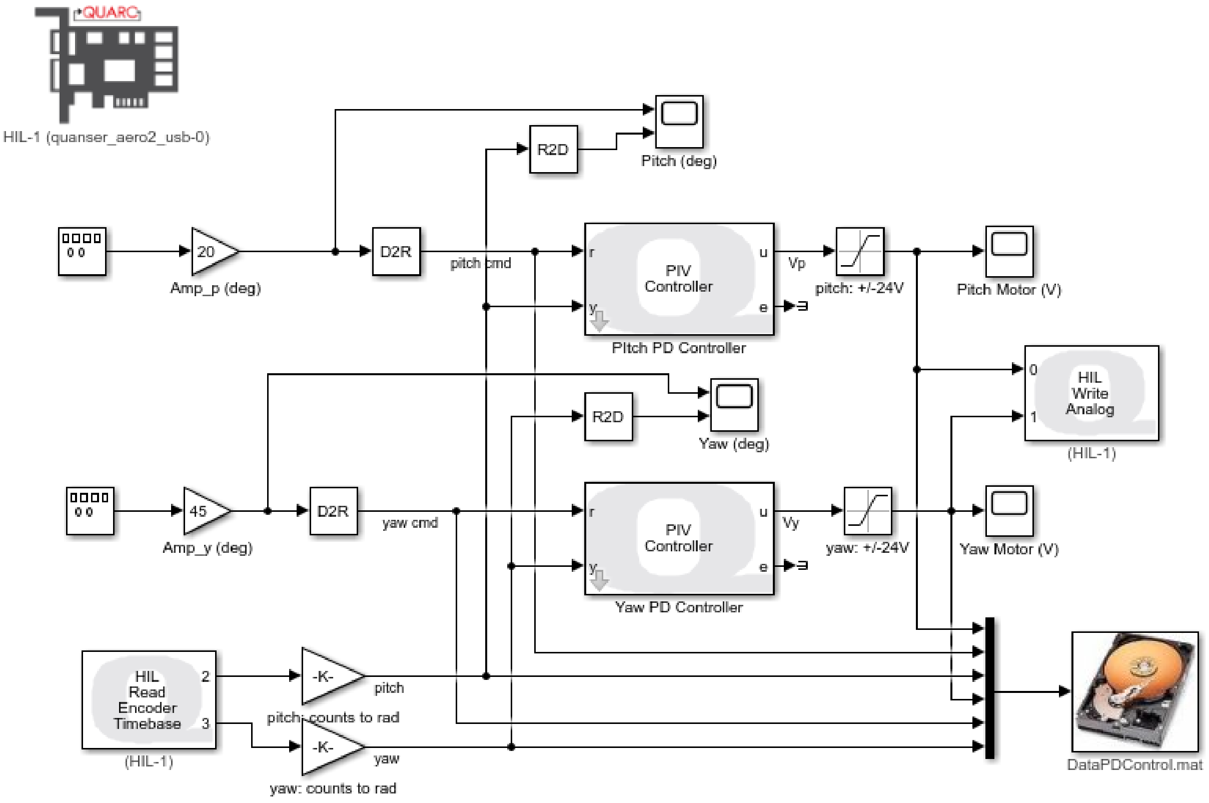


Figure 4: Simulink model used with QUARC to run the PD control on the Aero 2.

The Simulink model uses the `HIL Write Analog` and `HIL Read Timebase` blocks from the *QUARC Targets* library to apply the control input voltage to the rotors and measure the corresponding pitch and yaw response. The response is saved into a MATLAB *.mat files using the `To Host File` block. The PD control on each axis is implemented using the QUARC *PIV Controller* block.

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. **Unlock** the pitch axis and **lock** the yaw axis.
5. Rear rotor 1 is **vertical** and front rotor 0 is **horizontal**.
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_2dof_pd.slx");
```

The scopes in Figure 5 show a sample pitch angle PD response.

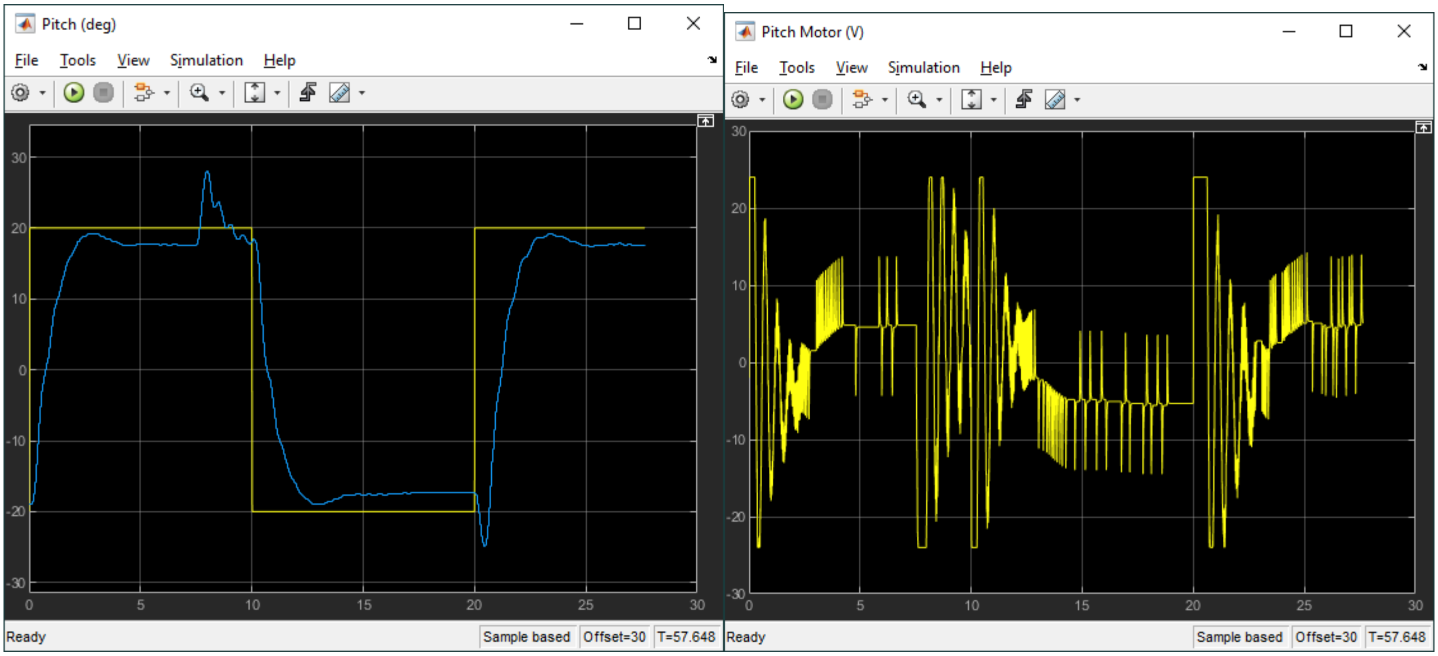


Figure 5 - Pitch PD control response

The scopes in Figure 6 show a sample yaw PD response.

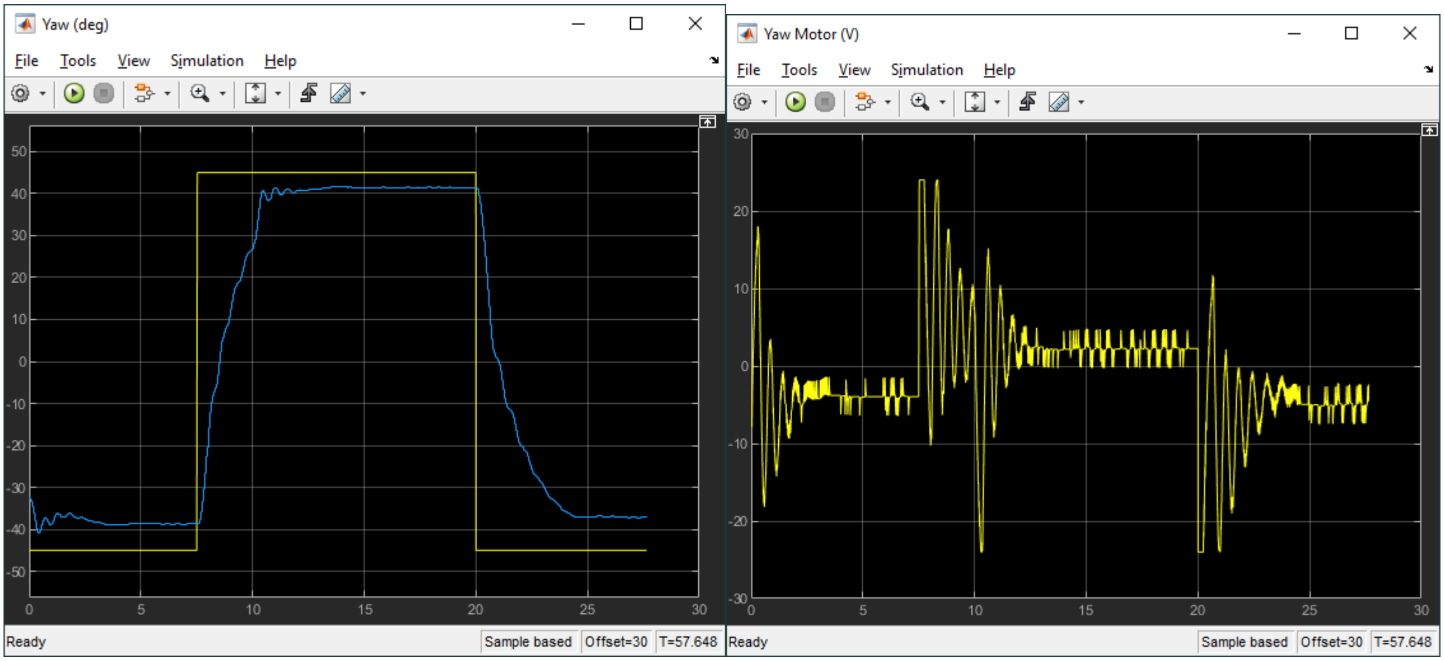


Figure 6 - Yaw PD control response.

Plot Response

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

```
t = DataPDControl(1,:); % time (s)
Vp = DataPDControl(2,:); % pitch voltage (V)
theta_d = DataPDControl(3,:); % desired pitch angle (rad)
theta = DataPDControl(4,:); % pitch angle (rad)
Vy = DataPDControl(5,:); % yaw voltage (V)
yaw_d = DataPDControl(6,:); % desired yaw angle (rad)
yaw = DataPDControl(7,:); % yaw angle (rad)
%
figure;
subplot(2,2,1);
plot(t,theta_d,t,theta);
title('Pitch');
ylabel('Angle (rad)');
subplot(2,2,3);
plot(t,Vp);
ylabel('Pitch Motor (V)');
xlabel('time (s)');
%
subplot(2,2,2);
plot(t,yaw_d,t,yaw);
title('Yaw');
ylabel('Angle (rad)');
subplot(2,2,4);
plot(t,Vy);
ylabel('Yaw Motor (V)');
xlabel('time (s)');
sgtitle('PD Control Hardware Response')
```

Looking at the pitch response when the step starts at 50 s, the peak time and overshoot are approximately:

$t_p = 3.41$ s and $PO = \dfrac{19.3 - 17.6}{40} \times 100 = 4.30\%$.

Based on the yaw response when the step start at 37.5 s, the peak time and overshoot are approximately:

$t_p = 3.50$ s and $PO = 0$

These measurements are taken on step responses that have minimal cross-coupling effect. Comparing these to the control requirements, the peak time of the pitch is slower than anticipated but all the other requirements are met. The pitch peak time could be further reduced by increasing the proportional gain but that may increase the overshoot over 5%.