Aero 2: 2 DOF Helicopter LQR Control Design

Concept Review

In this lab module, a state-feedback controller is designed to regulate the pitch and yaw angles of the Aero 2 Experiment to desired angles. By using the state-space model given above, we can find a control gain K based on the coupled dynamics to stabilize this system. The control gains are computed using the Linear-Quadratic Regulator (LQR) algorithm. The general state-feedback control is illustrated in Figure 1.

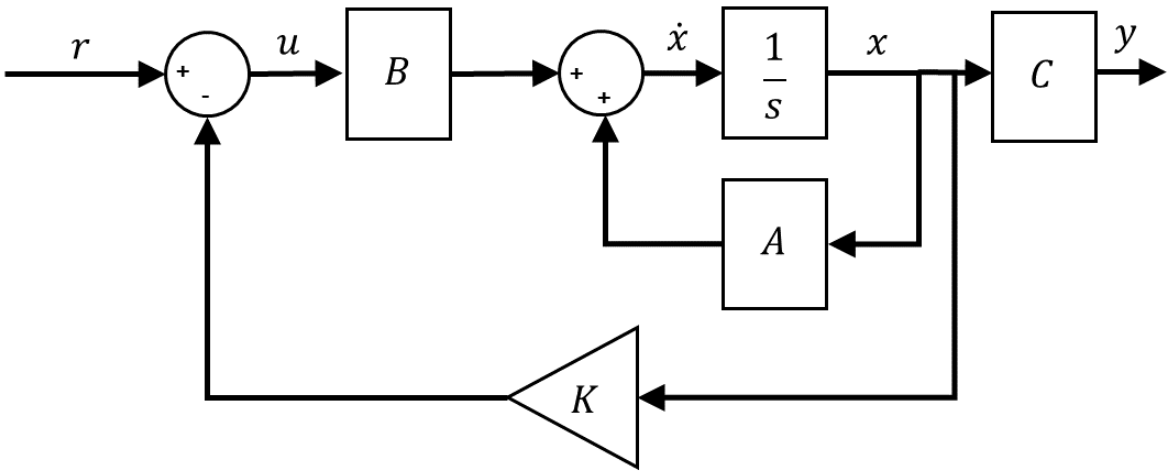


Figure 1: State-feedback control

The state-feedback controller is defined as

$$u = K(r - x),$$

where$x$ is the system state defined in the Aero 2 state-space model given in `Aero2_2DOF_Modeling.mlx`,

$$r^T = \begin{bmatrix} \theta_d & \psi_d & 0 & 0 \end{bmatrix}$$

is the reference command (or setpoint) state with the desired pitch and yaw angles, θ_d and ψ_d , and

$$u^T = \begin{bmatrix} V_p & V_y \end{bmatrix}$$

is the control input, where V_p is the front/pitch motor voltage and V_y is the tail or rear/yaw motor voltage.

The Linear Quadratic Regulator (LQR) optimization method can be used to find the control gain. Given the state-space model of the system, the LQR algorithm computes a control law u to minimize the performance criterion or cost function

$$J = \int_0^\infty (r(t) - x(t))^T Q(r(t) - x(t)) + u(t)^T R u(t) dt. \qquad (1)$$

The design matrices Q and R hold the penalties on the deviations of state variables from their setpoint and the control actions, respectively.

- When an element of Q is increased the cost function increases the penalty associated with any deviations from the desired setpoint of that state variable, and thus the specific control gain will be larger.
- When the values of the R matrix are increased, a larger penalty is applied to the aggressiveness of the control action, and the control gains are uniformly decreased.

Given the Aero 2 state-space model has four states and two control inputs: $Q \in \Re^{4x4}$, $R \in \Re^{2x2}$, and the control gain $K \in \Re^{2\times4}$. With the reference state, r, defined above, the controller can be expanded into the form

$$u = \begin{bmatrix} V_p \\ V_y \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} (r - x) = \begin{bmatrix} k_{11}(\theta_d - \theta) - k_{12}\alpha - k_{13}\dot{\theta} - k_{14}\dot{\alpha} \\ -k_{21}\theta + k_{22}(\psi_d - \psi) - k_{23}\dot{\theta} - k_{24}\dot{\alpha} \end{bmatrix}$$

Designing an LQR controller:

The MATLAB `lqr` command will be used to generate the state-feedback control gain K. In order for the closed-loop response to satisfy certain time-domain specifications, the closed-loop system is typically simulated using its dynamic model, in software, first. This is an iterative process. By adjusting the weighting matrices Q and R, and then running the simulation, we can find a control that satisfies the s requirements. Further, we must ensure that the control signal u is smooth (i.e., does not chatter) and does not saturate the actuator, i.e., within the limits of the motor.

State-Feedback LQR 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.

The LQR Q and R matrices can be tuned to match these specifications in simulation. This is an iterative process.

Load the State-space model

Load the AERO 2 Parameters

```
aero2_parameters;
```

Load stiffness, damping, and thrust parameters values found in the *System Identification* labs.

```
aero2_parameters_id;
```

Create state-space model in MATLAB

```
A = [0 0 1 0;
    0 0 0 1;
    -Ksp/Jp 0 -Dp/Jp 0;
    0 0 0 -Dy/Jy];

B = [0 0;
    0 0;
    Dt*Kpp/Jp Dt*Kpy/Jp;
    Dt*Kyp/Jy Dt*Kyy/Jy];

C = eye(2,4);
D = zeros(2,2);
aero2_ss = ss(A,B,C,D)
```

```
aero2_ss =

  A =
           x1        x2        x3        x4
   x1       0         0         1         0
   x2       0         0         0         1
   x3  -0.3208        0  -0.08582         0
   x4       0         0         0  -0.08064

  B =
           u1        u2
   x1       0         0
   x2       0         0
   x3  0.02318  0.009892
   x4 -0.02243   0.04289

  C =
      x1  x2  x3  x4
   y1  1   0   0   0
   y2  0   1   0   0

  D =
      u1  u2
   y1  0   0
   y2  0   0

Continuous-time state-space model.
```

```
eig(aero2_ss)
```

```
ans = 4×1 complex
  -0.0429 + 0.5648i
  -0.0429 - 0.5648i
   0.0000 + 0.0000i
  -0.0806 + 0.0000i
```

State-Feedback LQR Control Design

```
Q = diag([150 75 0 0 ]);
R = 0.01*eye(2,2);
K = lqr(A,B,Q,R)
```

```
K = 2×4
   99.5844  -37.6373   80.6953  -24.3159
   47.5974   77.9964   40.6478   52.9117
```

LQR Control Simulation

The `s_aero2_2dof_lqr` Simulink model shown in Figure 2 can be used to simulate the pitch and yaw state-feedback control of the Aero 2 - 2 DOF Helicopter. The *State-Space* block is used to model the 2 DOF Helicopter the model includes the motor limits of ± 24V to represent the system hardware more accurately. The model includes coupled dynamics due to the reaction torque.

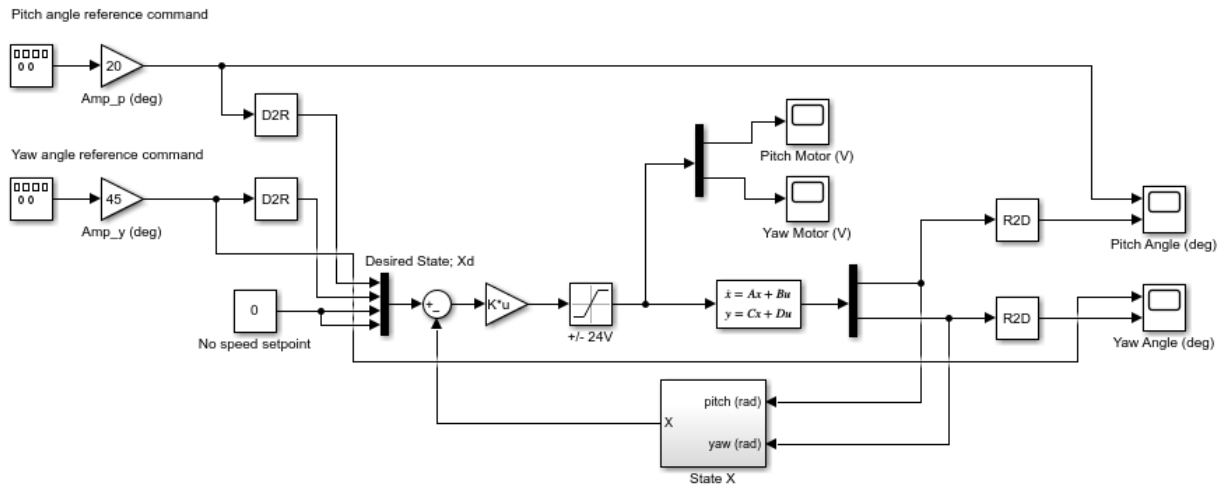


Figure 2: Simulates the LQR control of the pitch and yaw axes on the 2 DOF Aero 2.

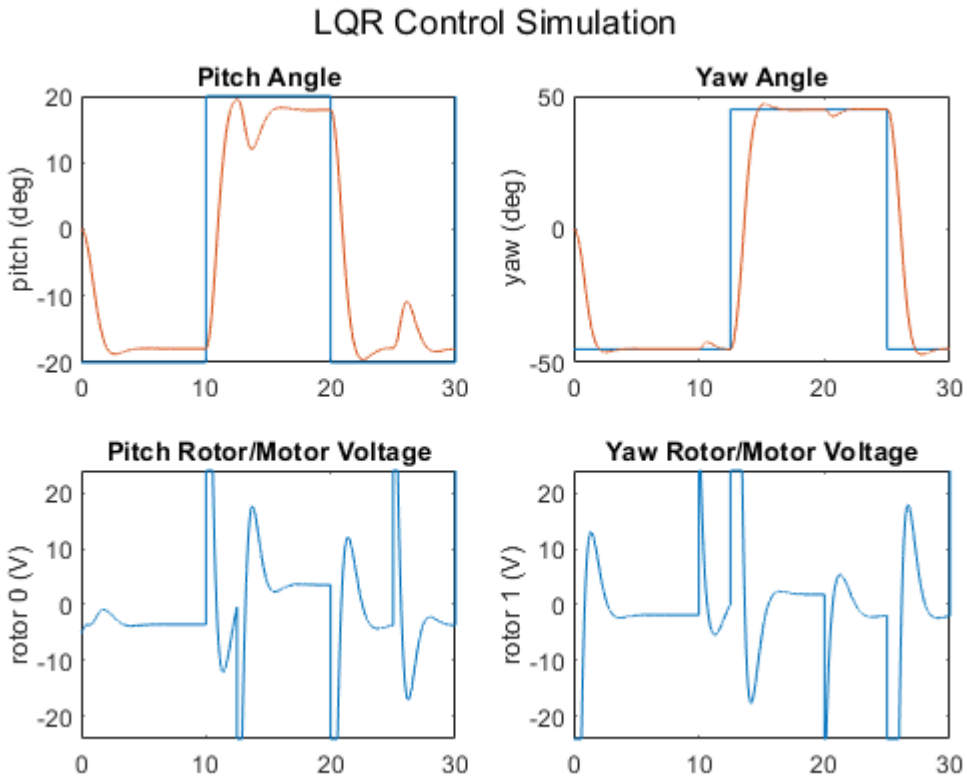
Run the Simulink simulation and verify that the simulated response satisfies the design specifications.

```
open("s_aero2_2dof_lqr_control")
sim("s_aero2_2dof_lqr_control")
```

Plot response using the data logged from the Simulink scopes.

```
t = Aero2_LQR_Sim_Vp.time; % time (s)
Vp = Aero2_LQR_Sim_Vp.signals.values; % pitch voltage (V)
theta_d_sim = Aero2_LQR_Sim_Pitch.signals(1).values; % desired/command pitch angle (rad)
theta_sim = Aero2_LQR_Sim_Pitch.signals(2).values; % pitch angle response (rad)
Vy = Aero2_LQR_Sim_Vy.signals.values; % yaw voltage (V)
psi_d_sim = Aero2_LQR_Sim_Yaw.signals(1).values; % desired/command yaw angle (rad)
psi_sim = Aero2_LQR_Sim_Yaw.signals(2).values; % yaw angle response (rad)
%
figure;
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('LQR Control Simulation');
```



Simulated Response Results

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

$t_p = 2.5$ s and $PO = \frac{19.52 - 17.92}{17.92 - (-17.92)} \times 100 = 4.46\%$.

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

$t_p = 2.71$ s and $PO = \frac{47.05 - 45}{90} \times 100 = 2.28\%$.

The peak time and percentage overshoot for both the pitch and the yaw satisfy the specifications. The controller does saturate at the ± 24 motor limits but only for a short duration. Given that specifications are satisfied, this LQR control can be tested on the Aero 2 hardware.

Running LQR Control on the Hardware

The `q_aero2_2dof_lqr` Simulink model shown in Figure 3 will be used with QUARC to run the LQR-based state-feedback control on the Aero 2.

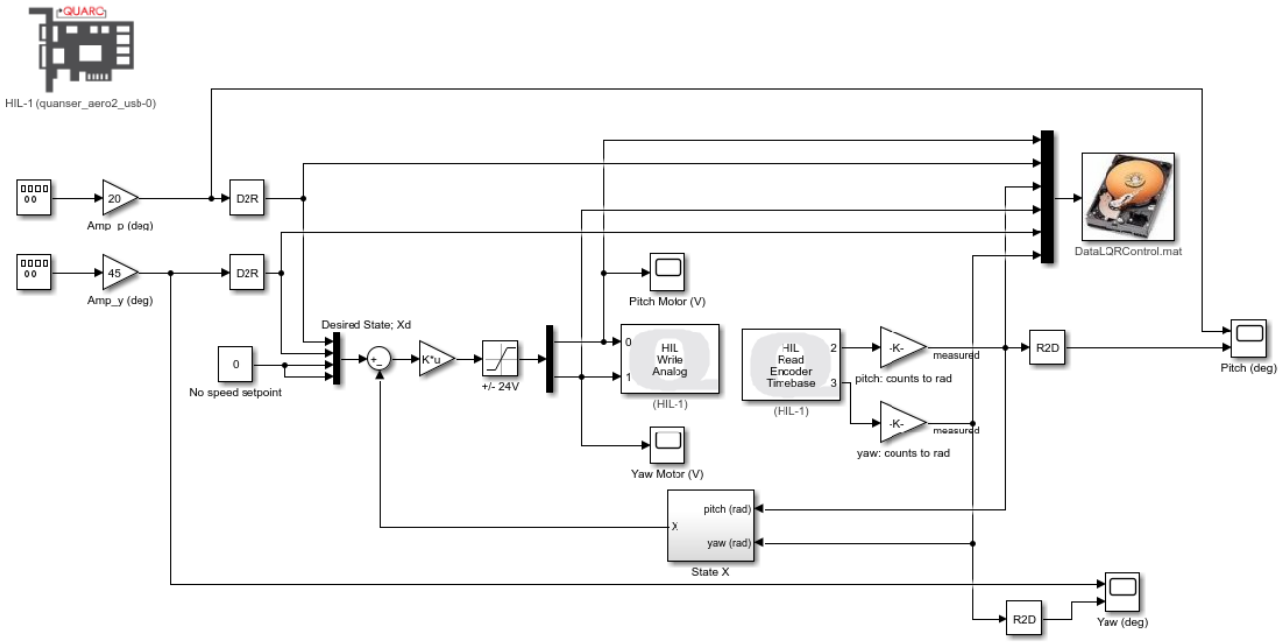


Figure 3: Simulink model used with QUARC to run the state-feedback control on the Aero 2.

The Simulink model uses the `HIL Write Analog` and `HIL Read Timebase` blocks from the *QUARC Targets* library to 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.

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_lqr_control.slx");
```

The scopes in Figure 4 show a sample pitch angle PD response.

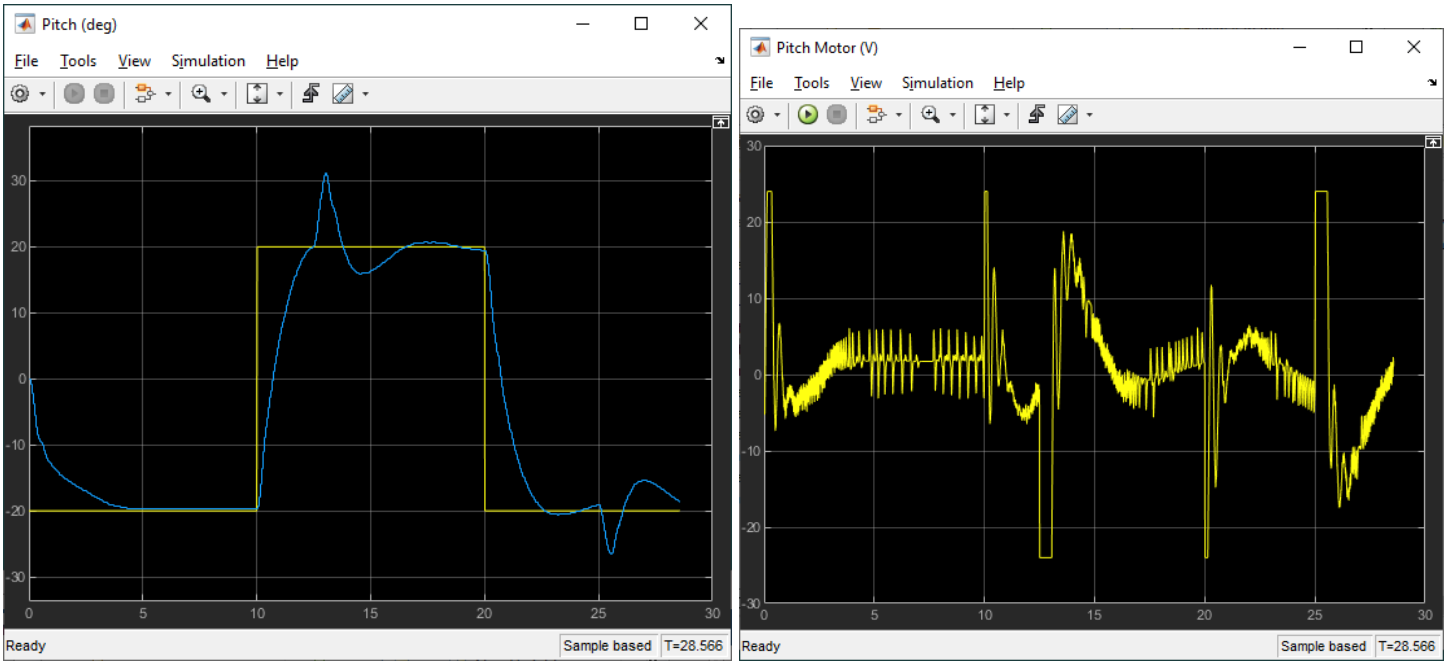


Figure 4 - Pitch LQR control response

The scopes in Figure 5 show a sample yaw LQR response.

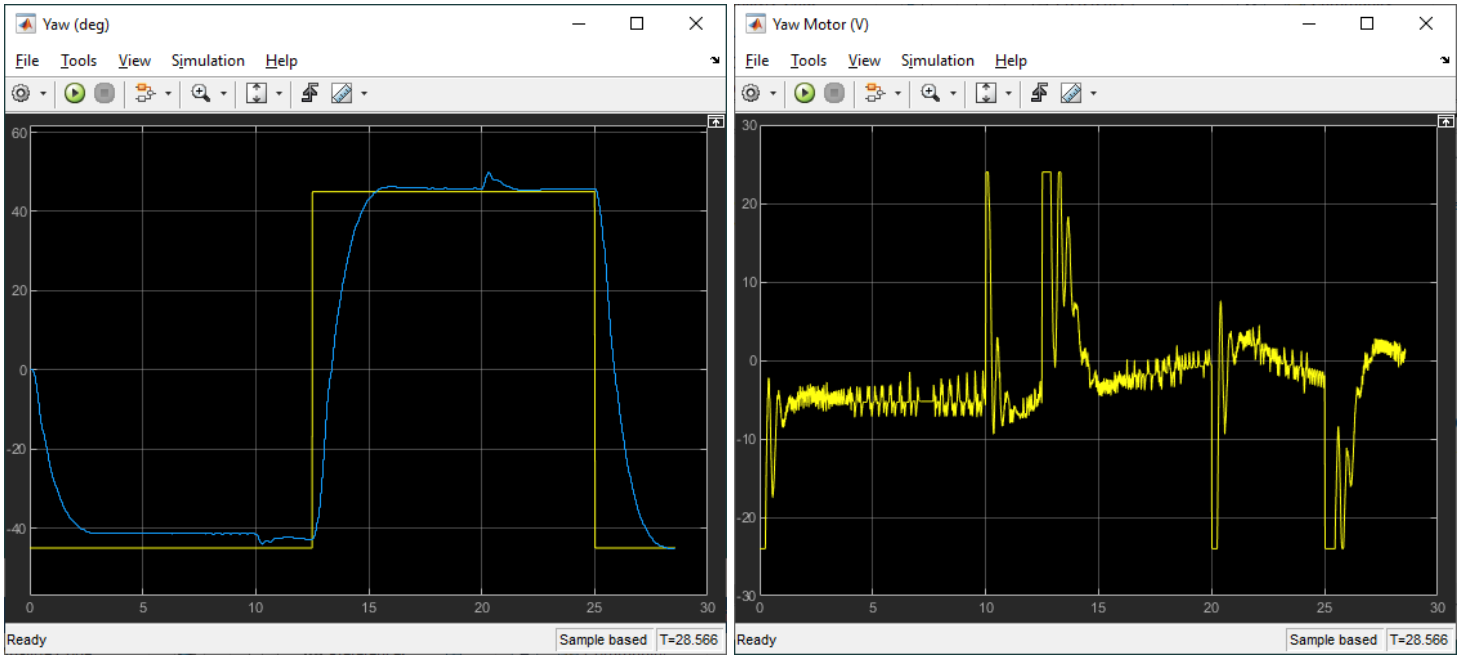
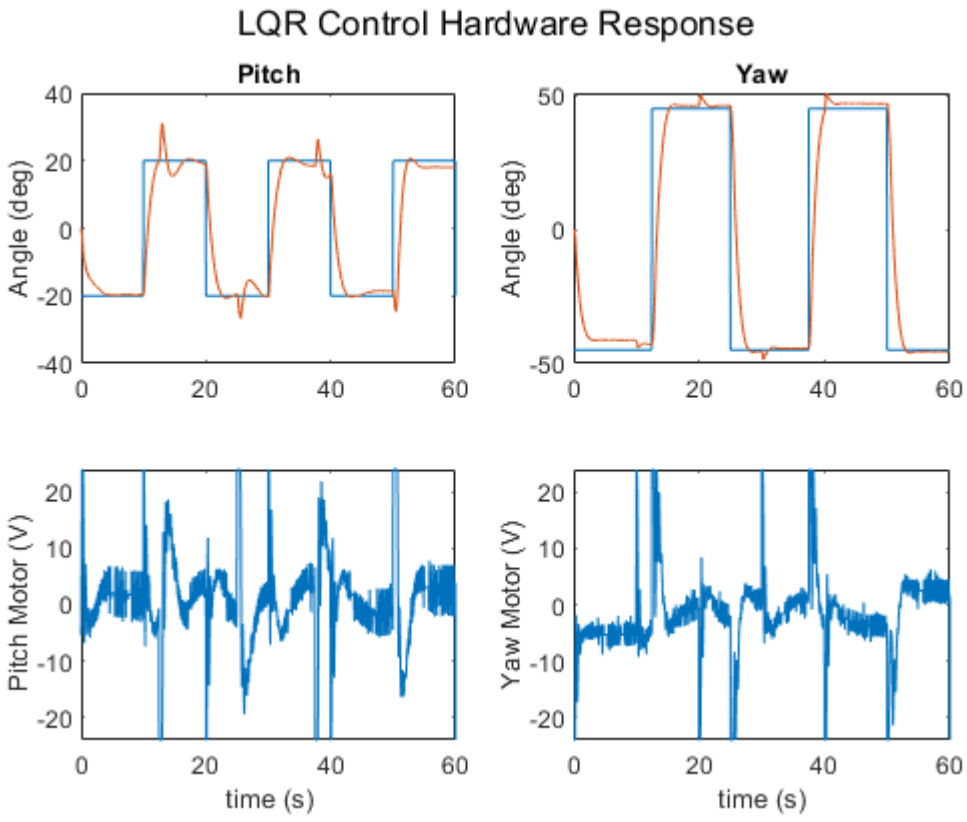


Figure 5 - Yaw LQR control response.

Plot Response

```
% Load measured data from past run
load('DataLQRControl.mat');
% store in variables
t = DataLQRControl(1,:); % time (s)
Vp = DataLQRControl(2,:); % pitch voltage (V)
```

```
theta_d = 180/pi*DataLQRControl(3,:); % desired pitch angle (deg)
theta = 180/pi*DataLQRControl(4,:); % pitch angle (deg)
Vy = DataLQRControl(5,:); % yaw voltage (V)
yaw_d = 180/pi*DataLQRControl(6,:); % desired yaw angle (deg)
yaw = 180/pi*DataLQRControl(7,:); % yaw angle (deg)
%
figure;
subplot(2,2,1);
plot(t,theta_d,t,theta);
title('Pitch');
ylabel('Angle (deg)');
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 (deg)');
subplot(2,2,4);
plot(t,Vy);
ylabel('Yaw Motor (V)');
xlabel('time (s)');
sgtitle('LQR Control Hardware Response');
```



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

$t_p = 52.21 - 50.00 = 2.21$ s and $PO = \frac{20.9 - 20.4}{40} \times 100 = 1.25\%$.

Based on the yaw response when the step starts at 12.5 s, the peak time and overshoot are approximately:

$t_p = 15.53 - 12.50 = 3.03$ s and $PO = 100 \times \frac{48.5 - 47.0}{90} = 1.67\%$.

Both the pitch and yaw response satisfy the control requirements.

LQR Control vs. PD Control

Compared to the PD response, the state-feedback control has less steady-state error and is able to compensate for the reaction torques more effectively. This can be attributed to having a control design that is based on a model that includes the reaction torque terms $K_{py}D_tV_y$ and $K_{yp}D_tV_p$. As shown in the expanded controller, the state-feedback control has more gains acting on all our state for the pitch and yaw axes. The PD control was a decoupled control design that considered the pitch-axis and yaw-axis separately.