Multicopter Design and Control Practice
—— A Series Experiments Based on MATLAB and Pixhawk

Lesson 09 Attitude Controller Design Experiment

Quan Quan, Associate Professor, qq_buaa@buaa.edu.cn
School of Automation Science and Electrical Engineering,
BeihangUniversity, Beijing 100191, China.

Outline

1. Preliminary
2. Basic Experiment
3. Analysis Experiment
4. Design Experiment
5. Summary

Preliminary

Basic concepts

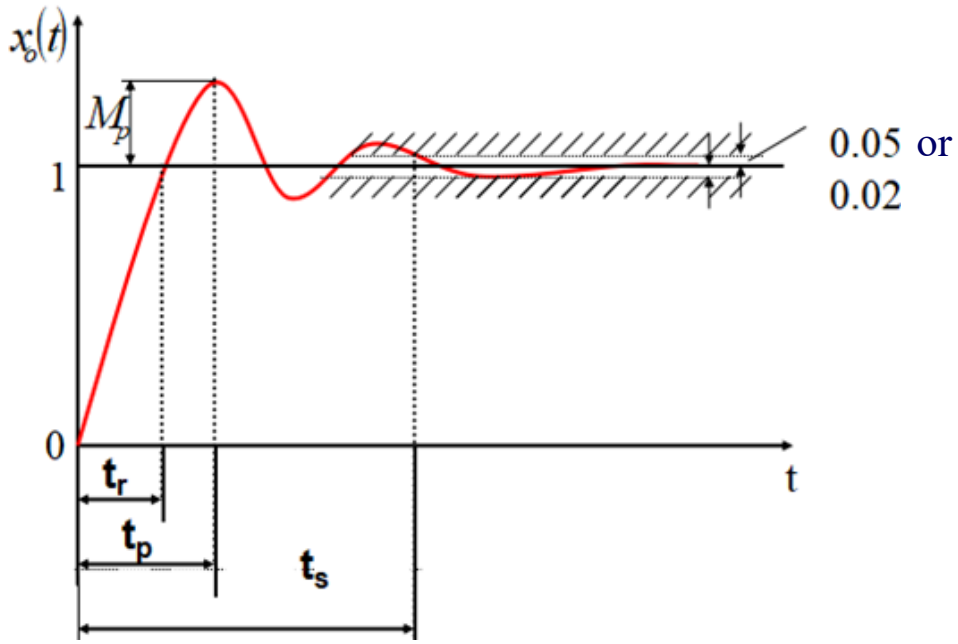


Figure. the step response of the second-order system

(1) System characteristics in time domain

For Second-order system $G(s)=\dfrac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$

where $0<\xi<1$, the step response is shown on the left.

1) overshoot $M_p=\dfrac{x_o(t_p)-x_o(\infty)}{x_o(\infty)}*100\%=e^{-\xi\pi/\sqrt{1-\xi^2}}*100\%$

2) setting time

when $\xi<0.8$, the settling time is

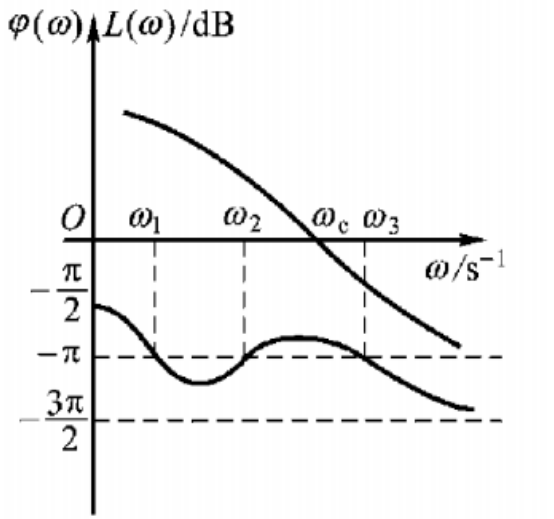
$$t_s=\frac{3.5}{\xi\omega_n} \quad \text{or} \quad t_s=\frac{4.5}{\xi\omega_n}$$

Preliminary

Basic concepts

(2) Bode plot and steady-state error

Bode plot draws the characteristics of open-loop amplitude and phase on logarithmic coordinates. The logarithm stability criterion judges the stability of the closed-loop system according to the relationship between the open-loop logarithm amplitude frequency and the logarithm phase frequency curve.



Figure： Stability Margin

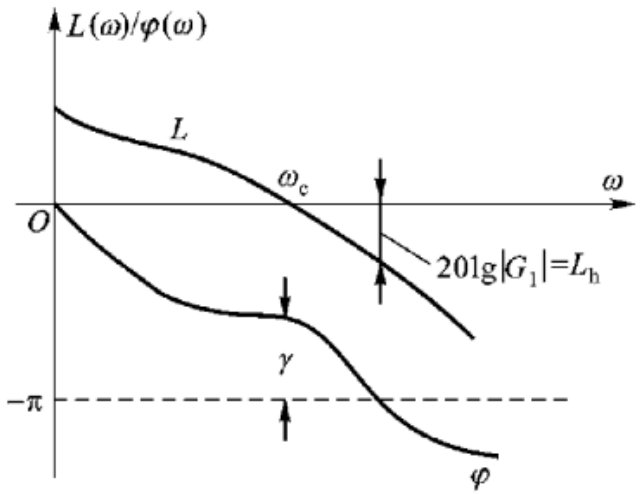


Figure. logarithm phase frequency curve

Basic concepts

(2) Bode plot and steady-state error

Phase margin γ: when $L(\omega)=0\,\mathrm{dB}$, the difference value between the phase curve and $-\pi$, i.e.,

$$\gamma=\angle G(j\omega_c)H(j\omega_c)-(-180^\circ) \qquad L(\omega_c)=0\,\mathrm{dB}$$

Where ω_c represents the cut-off frequency.

Gain Margin h: When $\angle G(j\omega_1)H(j\omega_1) = -\pi$

$$h(\mathrm{dB})=20\lg\left|\frac{1}{G(j\omega_1)H(j\omega_1)}\right|=-20\lg\left|G(j\omega_1)H(j\omega_1)\right|$$

When the closed-loop system is stable, the larger the phase and gain margin is, the more stable the system is. The stability margin also reflects the smoothness of the system, such as the overshoot and so on. General requirements:

$$\gamma>40^\circ$$

$$h>6\mathrm{dB}$$

Preliminary

Framework of Low-Level Flight Control of Multicopters

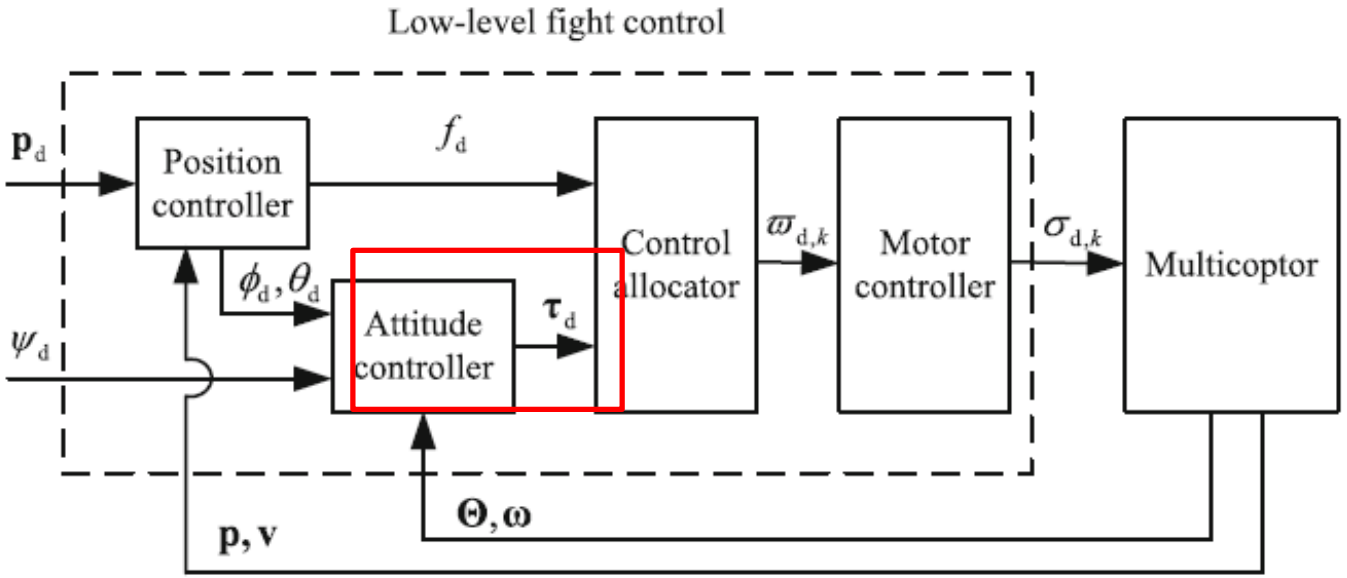


Figure. Closed-loop structure of a low-level fight control system for multicopters

The multicopter is underactuated because the system has six outputs (position $\mathbf{p} \in R^3$ and attitude $\mathbf{\Theta} \in R^3$), but only four independent inputs (total thrust $f \in R$ and three-axis moment $\boldsymbol{\tau} \in R^3$). In the design of multicopter flight controller, the control strategy of inner and outer loops can be used, in which the inner loop controls the attitude angle, while the outer loop controls the position. The inner and outer loop control is used to realize the lifting, hovering, side flying and other flight modes of the multicopter.

Preliminary

Attitude Control

For multicopters, the attitude control is the basis of the position control. At present, the commonly-used attitude representations for a rigid-body include the Euler angles and rotation matrix.

Table: Comparison of two attitude representations

Attitude representations	Advantages	Disadvantages
Euler angles	No redundant parameter; Clear physical meaning	Singularity problem when the pitch angle is 0; Plenty of transcendental function operation; Gimbal Lock
Rotation matrix	No singularity problem; No transcendental function operation; Applicability for the continuous rotation; Global and unique; Easy to interpolate.	Six redundant parameters

Preliminary

Attitude control

The attitude control objective is: given the desired attitude angle $\boldsymbol{\Theta}_{\mathrm{d}} \triangleq \left[\boldsymbol{\Theta}_{\mathrm{hd}}^{\mathrm{T}} \quad \psi_{\mathrm{d}}\right]^{\mathrm{T}}$,design a controller $\boldsymbol{\tau} \in \mathbb{R}^3$ such that $\lim_{t\to\infty}\left\|\mathbf{e}_{\boldsymbol{\Theta}}(t)\right\| = 0$,where $\mathbf{e}_{\boldsymbol{\Theta}} \triangleq \boldsymbol{\Theta} - \boldsymbol{\Theta}_{\mathrm{d}}$. Here, $\boldsymbol{\Theta}_{\mathrm{hd}}$ is generated by the position controller and ψ_{d} is given by the mission planner. To achieve that, according to

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{\omega}$$

The desired angle velocity $\boldsymbol{\omega}_{\mathrm{d}}$ is designed as

$$\boldsymbol{\omega}_{\mathrm{d}} = -\mathbf{K}_{\boldsymbol{\Theta}}\mathbf{e}_{\boldsymbol{\Theta}}$$

Where $\mathbf{K}_{\boldsymbol{\Theta}} \geq 0$. Then according to

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau}$$

The PID controller is designed as

$$\boldsymbol{\tau}_{\mathrm{d}} = -\mathbf{K}_{\boldsymbol{\omega}\mathrm{p}}\mathbf{e}_{\boldsymbol{\omega}} - \mathbf{K}_{\boldsymbol{\omega}\mathrm{i}}\int \mathbf{e}_{\boldsymbol{\omega}} - \mathbf{K}_{\boldsymbol{\omega}\mathrm{d}}\dot{\mathbf{e}}_{\boldsymbol{\omega}}$$

Where $\mathbf{e}_{\boldsymbol{\omega}} \triangleq \boldsymbol{\omega} - \boldsymbol{\omega}_{\mathrm{d}}$, $\mathbf{K}_{\boldsymbol{\omega}\mathrm{p}}, \mathbf{K}_{\boldsymbol{\omega}\mathrm{i}}, \mathbf{K}_{\boldsymbol{\omega}\mathrm{d}} \in \mathbb{R}^{3\times 3}$

Preliminary

The control system compensation

The following mainly introduces the series compensation. The structure of the system with series compensation is shown in the figure. Where Gc(s) is the transfer function of the series compensator, and G(s) is the transfer function of the invariant part of the system. In engineering practice, the commonly used series compensation includes lead compensation, lag compensation and lag lead compensation.

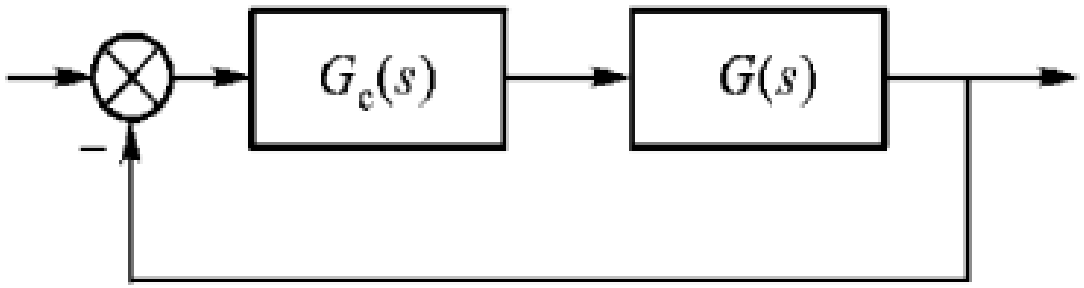


Figure. System with series compensation

Preliminary

The control system compensation

(1) Phase-Lead compensation

$$G_c(s) = \frac{1+aTs}{1+Ts}(a>1)$$

Phase-lead compensation occurs in $\left(\frac{1}{aT}, \frac{1}{T}\right)$

The maximum lead-phase is $\varphi_m = \arcsin\frac{a-1}{a+1}$

This maximum value occurs at the geometric center of the logarithmic frequency characteristic curve, and the corresponding angular frequency is:

$$\omega_m = \frac{1}{\sqrt{a}T}$$

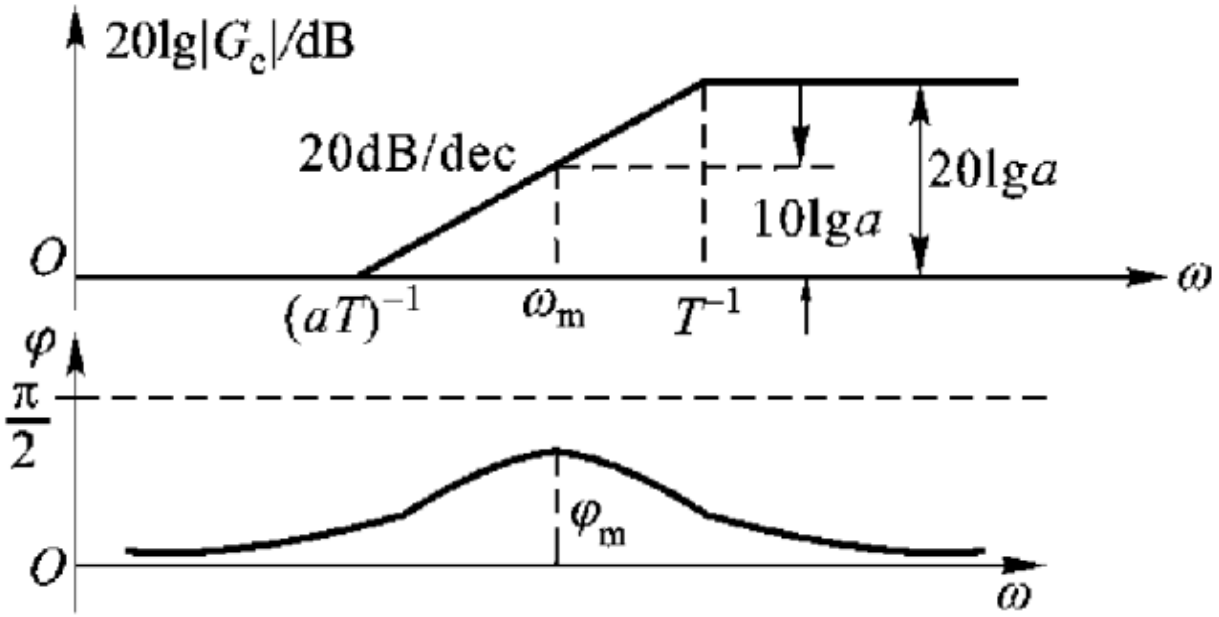


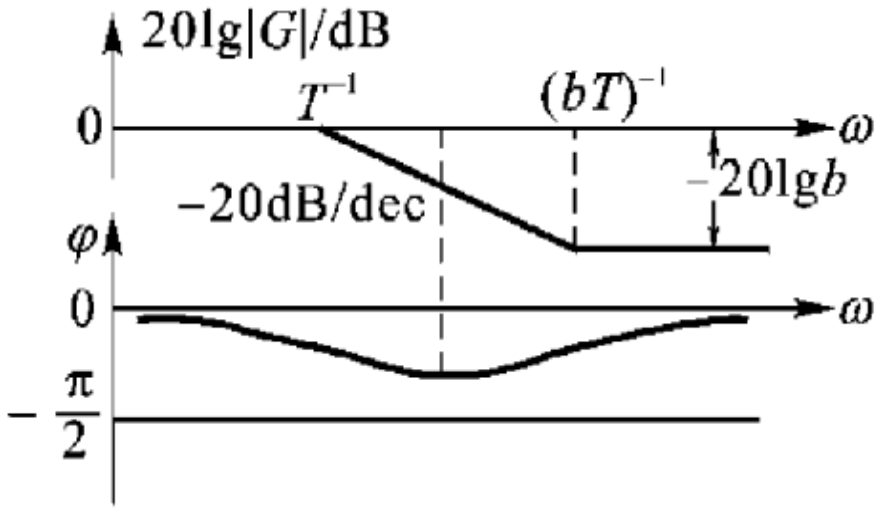
Figure. Phase-Lead compensation curve

Preliminary

The control system compensation

(2) Phase-lag compensation

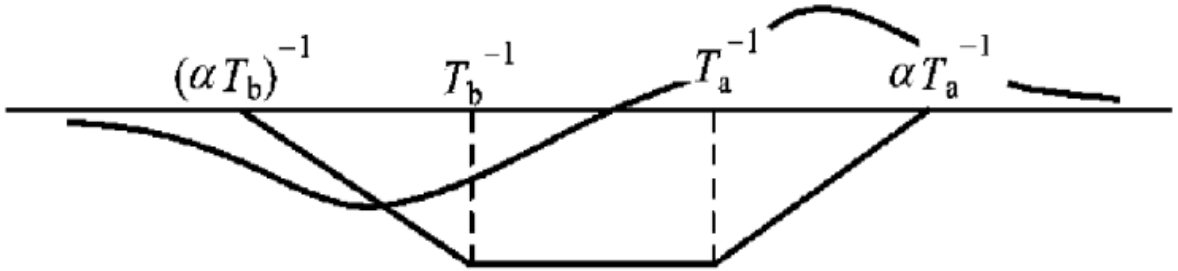
$$G_c(s) = \frac{1+bTs}{1+Ts}(b<1)$$



(3) Phase lead-and-lag compensation

$$G_c(s) = \frac{(1+bT_1s)(1+aT_2s)}{(1+T_1s)(1+T_2s)}$$

Where $a>1, b>1$**,and** $bT_1 > aT_2$



Preliminary

Control Allocation

Briefly, the control allocation problem can be described as:

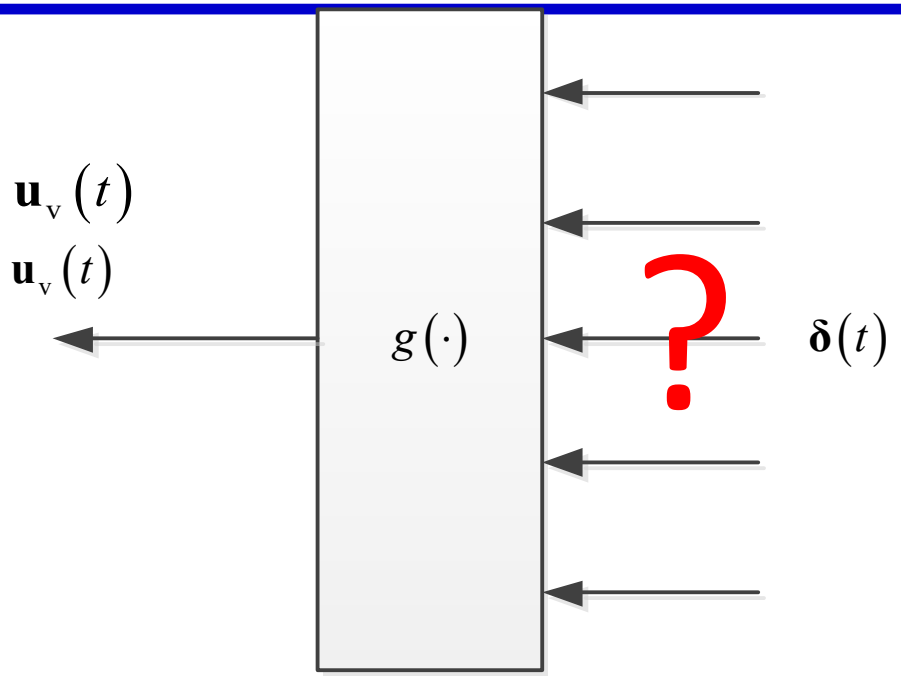
given $\boldsymbol{\delta}(t)$ to satisfy

$$\mathbf{u}_{\mathrm{v}}(t) = g\left(\boldsymbol{\delta}(t)\right)$$

Where $g:\mathbb{R}^m \to \mathbb{R}^n$ is the mapping from control inputs of actuators in controlled systems to the pseudo control command. In particular, the linear control allocation problem is based on

$$\mathbf{u}_{\mathrm{v}}(t) = \mathbf{B}\boldsymbol{\delta}(t)$$

Where $\mathbf{B} \in \mathbb{R}^{m\times n}$ is the known control efficiency matric.

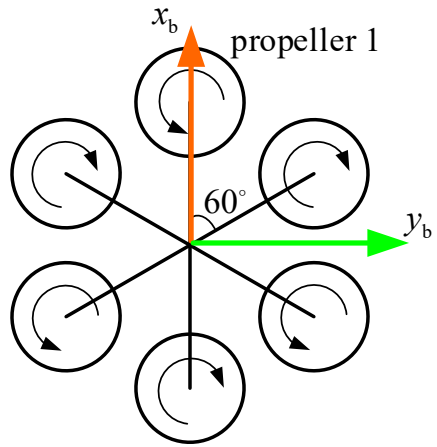


- **Separation of upper control and lower control**
- **Effective allocation to prevent saturation**
- **Improve robustness to faults and damages through allocation**
- **….**

Preliminary

Control Allocation

The control effectiveness models for a quadcopter with the x-configuration



$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} c_T & c_T & c_T & c_T \\ -\frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T \\ \frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T \\ c_M & c_M & -c_M & -c_M \end{bmatrix}}_{M_4} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \varpi_3^2 \\ \varpi_4^2 \end{bmatrix}$$

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_T & 0 & 0 & 0 \\ 0 & dc_T & 0 & 0 \\ 0 & 0 & dc_T & 0 \\ 0 & 0 & 0 & c_M \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \varpi_3^2 \\ \varpi_4^2 \end{bmatrix}$$

$$\Rightarrow \begin{cases} \varpi_1^2 = \frac{f}{4c_T} - \frac{\sqrt{2}\tau_x}{4dc_T} + \frac{\sqrt{2}\tau_y}{4dc_T} + \frac{\tau_z}{4c_M} \\ \varpi_2^2 = \frac{f}{4c_T} + \frac{\sqrt{2}\tau_x}{4dc_T} - \frac{\sqrt{2}\tau_y}{4dc_T} + \frac{\tau_z}{4c_M} \\ \varpi_3^2 = \frac{f}{4c_T} + \frac{\sqrt{2}\tau_x}{4dc_T} + \frac{\sqrt{2}\tau_y}{4dc_T} - \frac{\tau_z}{4c_M} \\ \varpi_4^2 = \frac{f}{4c_T} - \frac{\sqrt{2}\tau_x}{4dc_T} - \frac{\sqrt{2}\tau_y}{4dc_T} - \frac{\tau_z}{4c_M} \end{cases}$$

The unknown parameters can be compensated by the controller, assuming that the outputs of the four channels of throttle, pitch, roll and yaw are respectively σ_1、σ_2、σ_3、σ_4

$$\Rightarrow \begin{cases} \varpi_1^2 = \sigma_1 - \sigma_3 + \sigma_2 + \sigma_4 \\ \varpi_2^2 = \sigma_1 + \sigma_3 - \sigma_2 + \sigma_4 \\ \varpi_3^2 = \sigma_1 + \sigma_3 + \sigma_2 - \sigma_4 \\ \varpi_4^2 = \sigma_1 - \sigma_3 - \sigma_2 - \sigma_4 \end{cases}$$

Preliminary

Control Allocation

The control effectiveness model:

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} c_{\mathrm{T}} & c_{\mathrm{T}} & \cdots & c_{\mathrm{T}} \\ -dc_{\mathrm{T}}\sin\varphi_1 & -dc_{\mathrm{T}}\sin\varphi_2 & \cdots & -dc_{\mathrm{T}}\sin\varphi_{n_{\mathrm{r}}} \\ dc_{\mathrm{T}}\cos\varphi_1 & dc_{\mathrm{T}}\cos\varphi_2 & \cdots & dc_{\mathrm{T}}\cos\varphi_{n_{\mathrm{r}}} \\ c_{\mathrm{M}}\delta_1 & c_{\mathrm{M}}\delta_2 & \cdots & c_{\mathrm{M}}\delta_{n_{\mathrm{r}}} \end{bmatrix}}_{\mathbf{M}_{n_{\mathrm{r}}}} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \vdots \\ \varpi_{n_{\mathrm{r}}}^2 \end{bmatrix}$$

For a quadcopter, $\mathbf{M}_4 \in \mathbb{R}^{4\times4}$ is nonsingular, the control allocation matrix can be directly obtained by matrix can be directly obtained by matrix inversion, namely

$$\mathbf{P}_4 = \mathbf{M}_4^{-1}$$

where $\mathbf{P}_4 \in \mathbb{R}^{4\times4}$ .The allocation is unique. Furthermore, if a multicopter has more than four propellers, the allocation may be non-unique. In open source autopilots, the control allocation matrix is often obtained by calculating the pseudo-inverse, that is

$$\mathbf{P}_{n_{\mathrm{r}}} = \mathbf{M}_{n_{\mathrm{r}}}^{\dagger}$$

Preliminary

Implementation of Control Allocation in Autopilots

After the desired thrust f_{d} and the desired moment $\boldsymbol{\tau}_{\mathrm{d}}$ have been obtained, the desired propeller angular speeds are obtained as follows:

$$\begin{pmatrix} \varpi_{\mathrm{d},1}^2 \\ \varpi_{\mathrm{d},2}^2 \\ \vdots \\ \varpi_{\mathrm{d},n_{\mathrm{r}}}^2 \end{pmatrix} = \mathbf{P}_{n_{\mathrm{r}}} \begin{pmatrix} f_{\mathrm{d}} \\ \boldsymbol{\tau}_{\mathrm{d}} \end{pmatrix}$$

?

Relative to the parameters

We can obtain the desired speed for each propeller $\varpi_{d,i}, i = 1, 2, \cdots, n_{\mathrm{r}}$

In the engineering practice, the parameters in $\mathbf{M}_{n_{\mathrm{r}}}$ are unknown. Under this condition, a question arises: how is the control allocation conducted?

Preliminary

☐ Implementation of Control Allocation in Autopilots

In order to answer this question, the control effectiveness matrix for a standard multicopter is defined as a function matrix that

$$\mathbf{M}_{n_r}\left(c_{\mathrm{T}},c_{\mathrm{M}},d\right)=\begin{bmatrix} c_{\mathrm{T}} & c_{\mathrm{T}} & \cdots & c_{\mathrm{T}} \\ -dc_{\mathrm{T}}\sin\varphi_1 & -dc_{\mathrm{T}}\sin\varphi_2 & \cdots & -dc_{\mathrm{T}}\sin\varphi_{n_r} \\ dc_{\mathrm{T}}\cos\varphi_1 & dc_{\mathrm{T}}\cos\varphi_2 & \cdots & dc_{\mathrm{T}}\cos\varphi_{n_r} \\ c_{\mathrm{M}}\sigma_1 & c_{\mathrm{M}}\sigma_2 & \cdots & c_{\mathrm{M}}\sigma_{n_r} \end{bmatrix}$$

Which satisfied

$$\mathbf{M}_{n_r}\left(c_{\mathrm{T}},c_{\mathrm{M}},d\right)=\mathbf{P}_{\mathrm{a}}\mathbf{M}_{n_r}\left(1,1,1\right)$$

Where $\mathbf{P}_{\mathrm{a}}=\mathrm{diag}\left(c_{\mathrm{T}} \quad dc_{\mathrm{T}} \quad dc_{\mathrm{T}} \quad c_{\mathrm{M}}\right)$.Thus, the following relation exists

$$\mathbf{M}_{n_r}^{\dagger}\left(c_{\mathrm{T}},c_{\mathrm{M}},d\right)=\mathbf{M}_{n_r}^{\dagger}\left(1,1,1\right)\mathbf{P}_{\mathrm{a}}^{-1}$$

Parameters known yet **Unknown parameters**

Preliminary

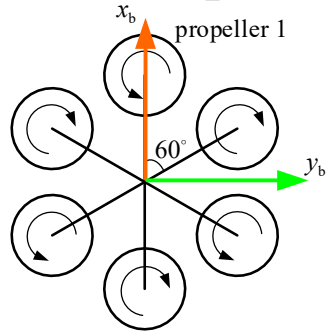
☐ Implementation of Control Allocation in Autopilots

Taking the hexacopter for example $\mathbf{M}_6(c_\mathrm{T}, c_\mathrm{M}, d)$ is

$$\mathbf{M}_6(c_\mathrm{T}, c_\mathrm{M}, d) = \begin{bmatrix} c_\mathrm{T} & c_\mathrm{T} & c_\mathrm{T} & c_\mathrm{T} & c_\mathrm{T} & c_\mathrm{T} \\ 0 & -\frac{\sqrt{3}dc_\mathrm{T}}{2} & -\frac{\sqrt{3}dc_\mathrm{T}}{2} & 0 & \frac{\sqrt{3}dc_\mathrm{T}}{2} & \frac{\sqrt{3}dc_\mathrm{T}}{2} \\ dc_\mathrm{T} & \frac{dc_\mathrm{T}}{2} & -\frac{dc_\mathrm{T}}{2} & -dc_\mathrm{T} & -\frac{dc_\mathrm{T}}{2} & \frac{dc_\mathrm{T}}{2} \\ c_\mathrm{M} & -c_\mathrm{M} & c_\mathrm{M} & -c_\mathrm{M} & c_\mathrm{M} & -c_\mathrm{M} \end{bmatrix}$$

$$\mathbf{M}_6^{\dagger}(1,1,1) = \frac{1}{6}\begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & -\sqrt{3} & 1 & -1 \\ 1 & -\sqrt{3} & -1 & 1 \\ 1 & 0 & -2 & -1 \\ 1 & \sqrt{3} & -1 & 1 \\ 1 & \sqrt{3} & 1 & -1 \end{bmatrix}$$

$$\begin{pmatrix} \varpi_{\mathrm{d},1}^2 \\ \varpi_{\mathrm{d},2}^2 \\ \vdots \\ \varpi_{\mathrm{d},6}^2 \end{pmatrix} = \mathbf{M}_6^{\dagger}(1,1,1)\mathbf{P}_\mathrm{a}^{-1}\begin{pmatrix} f_\mathrm{d} \\ \boldsymbol{\tau}_\mathrm{d} \end{pmatrix} = \mathbf{M}_6^{\dagger}(1,1,1)\begin{pmatrix} f_\mathrm{d}/c_\mathrm{T} \\ \tau_{\mathrm{d}x}/dc_\mathrm{T} \\ \tau_{\mathrm{d}y}/dc_\mathrm{T} \\ \tau_{\mathrm{d}z}/c_\mathrm{M} \end{pmatrix}$$



Preliminary

In order to make this chapter self-contained, the preliminary is from Chapter. 11 of “**Quan Quan. *Introduction to Multicopter Design and Control*. Springer, Singapore, 2017**”.

Basic Experiment

☐ Experimental Objective

■ Things to prepare

(1) Hardware: Multicopter System, Pixhawk Autopilot System;

(2) Software: MATLAB R2017b or above, Simulink-based Controller Design and Simulation Platform, HIL(Hardware in the loop) Simulation Platform, Instructional Package “e5.1” (https://rflysim.com/course).

■ Objectives

(1) Repeat the Simulink simulation of a quadcopter to analyze the functions of the control allocation;

(2) Record the step response of the attitude, obtain the open-loop Bode plot of the attitude control system and further analyze the stability margin of the closed-loop control system;

(3) Perform the HIL simulation.

Basic Experiment

☐ Experimental procedure

(1) Step1:SIL simulation – control allocation

1) Parameter Initialization

Run the file "e5\e5.1\Init_control.m" to initialize the parameters. Next, the Simulink file "AttitudeControl_Sim.slx" will open automatically as shown on the right.

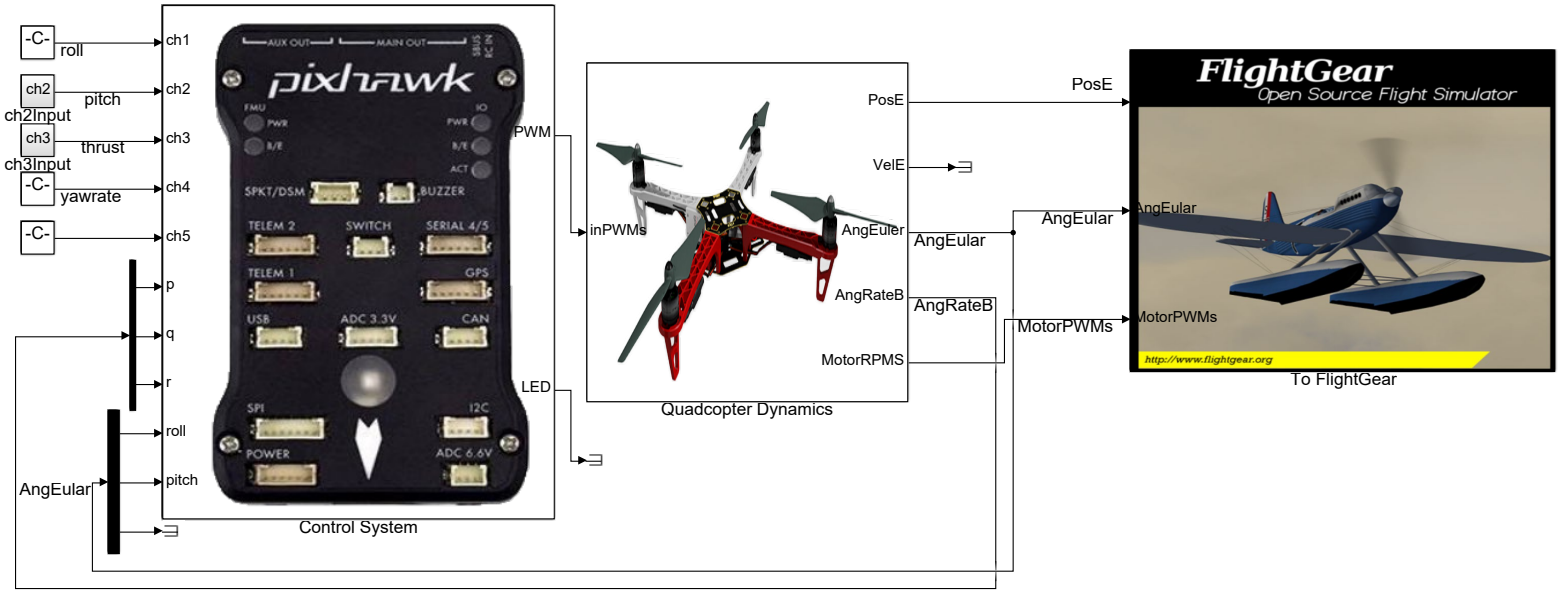


Figure. Control allocation testing, Simulink model "AttitudeControl_Sim.slx"

Basic Experiment

☐ Experimental procedure

2) Run the simulation

Open the file “FlightGear-F450” and click on the Simulink “Run” button to run “AttitudeControl_Sim.slx”. Subsequently, the motion of the quadcopter is observed in FlightGear, as shown on the right. The quadcopter in FlightGear climbs up for a short time, and then flies against the screen, corresponding to the $-o_b x_b$ axis.

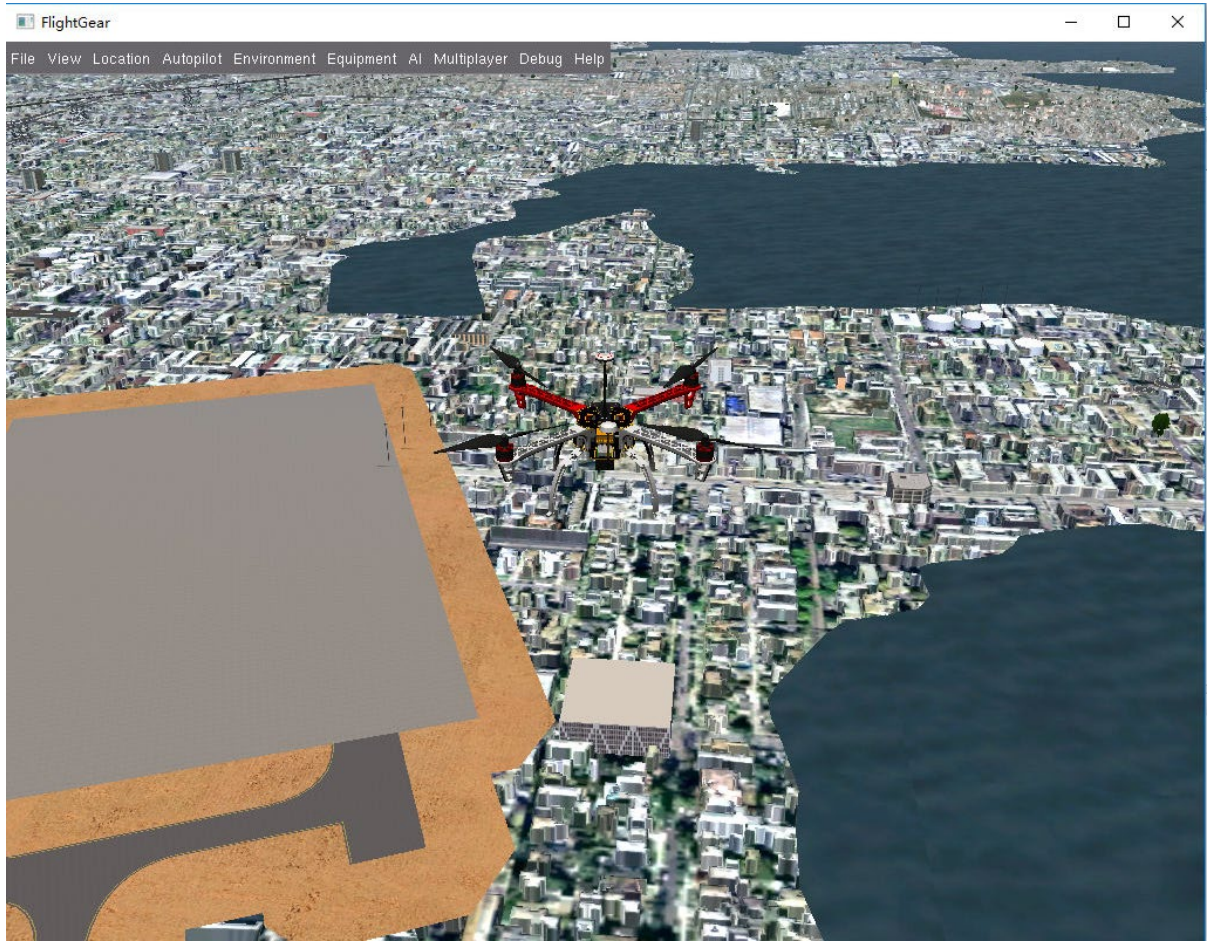


Figure. Quadcopter in FlightGear

Basic Experiment

☐ Experimental procedure

3) Simulation results

Set the input of the pitch channel of RC to 1600 with a delay time of 3 seconds.

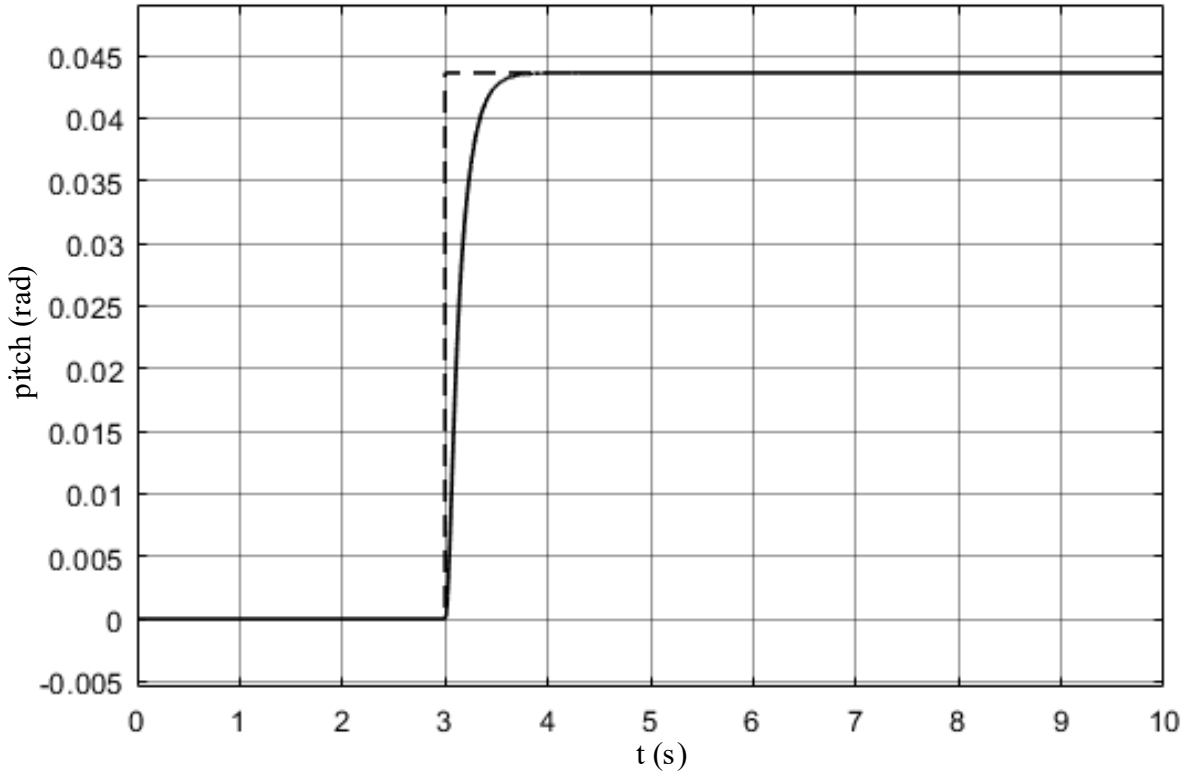


Figure. Step response of pitch angle

Basic Experiment

☐ Experimental procedure

4) Control allocation function

The control allocation module receives roll, pitch, yaw, and thrust control inputs and then assigns them to four motors, as shown on the right.

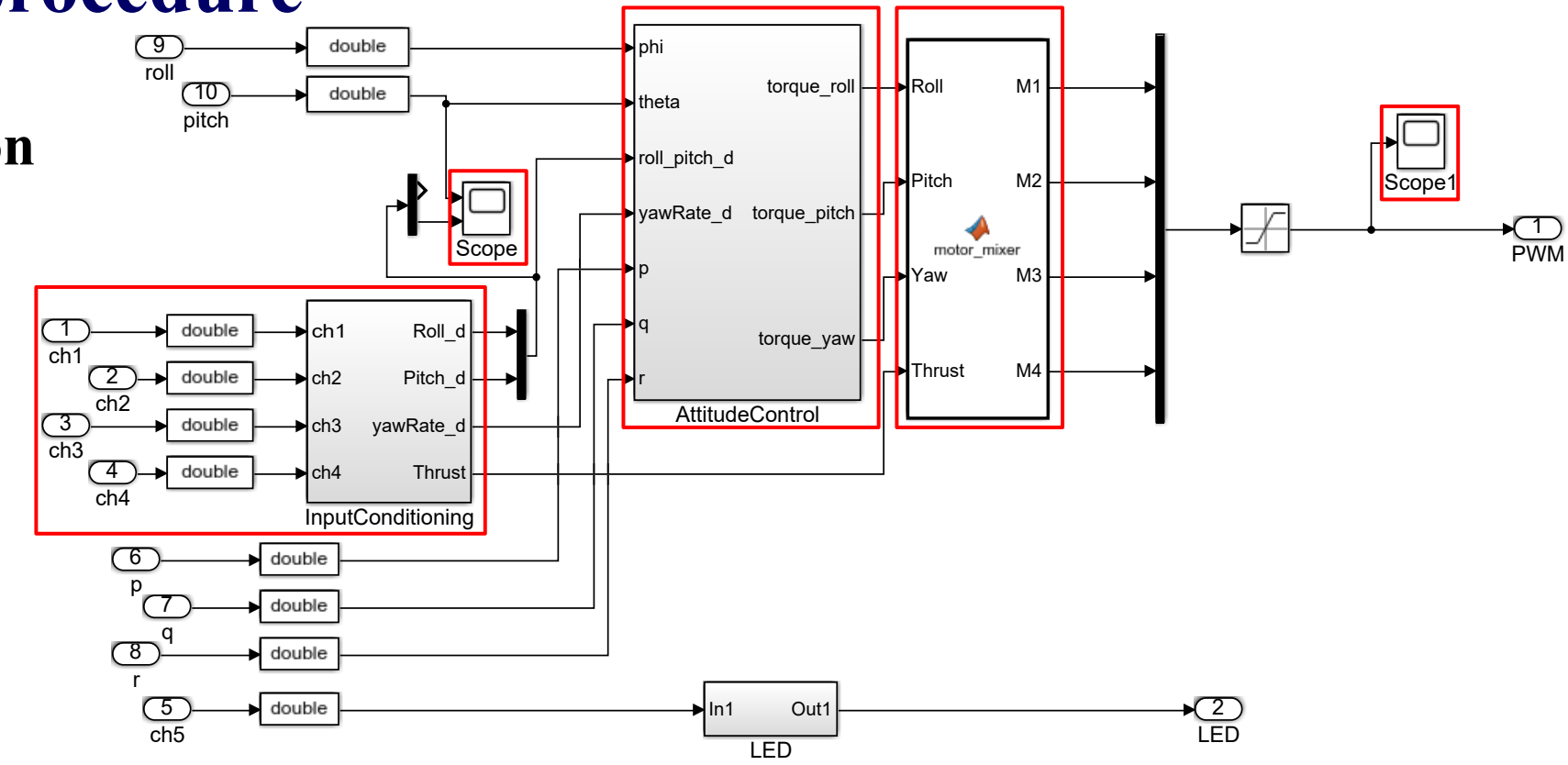


Figure. Structure of “Control System”module

Basic Experiment

☐ Experimental procedure

4) Control allocation function

When the pitch channel is given a value of 1600, the output of "Scope1" can be derived from the right picture. The control signals for motors #1 and #2 are increased , and the control signals for motors #3 and #4 are decreased.

$$\begin{cases} \varpi_1^2 = \sigma_1 - \sigma_3 + \sigma_2 + \sigma_4 \\ \varpi_2^2 = \sigma_1 + \sigma_3 - \sigma_2 + \sigma_4 \\ \varpi_3^2 = \sigma_1 + \sigma_3 + \sigma_2 - \sigma_4 \\ \varpi_4^2 = \sigma_1 - \sigma_3 - \sigma_2 - \sigma_4 \end{cases}$$

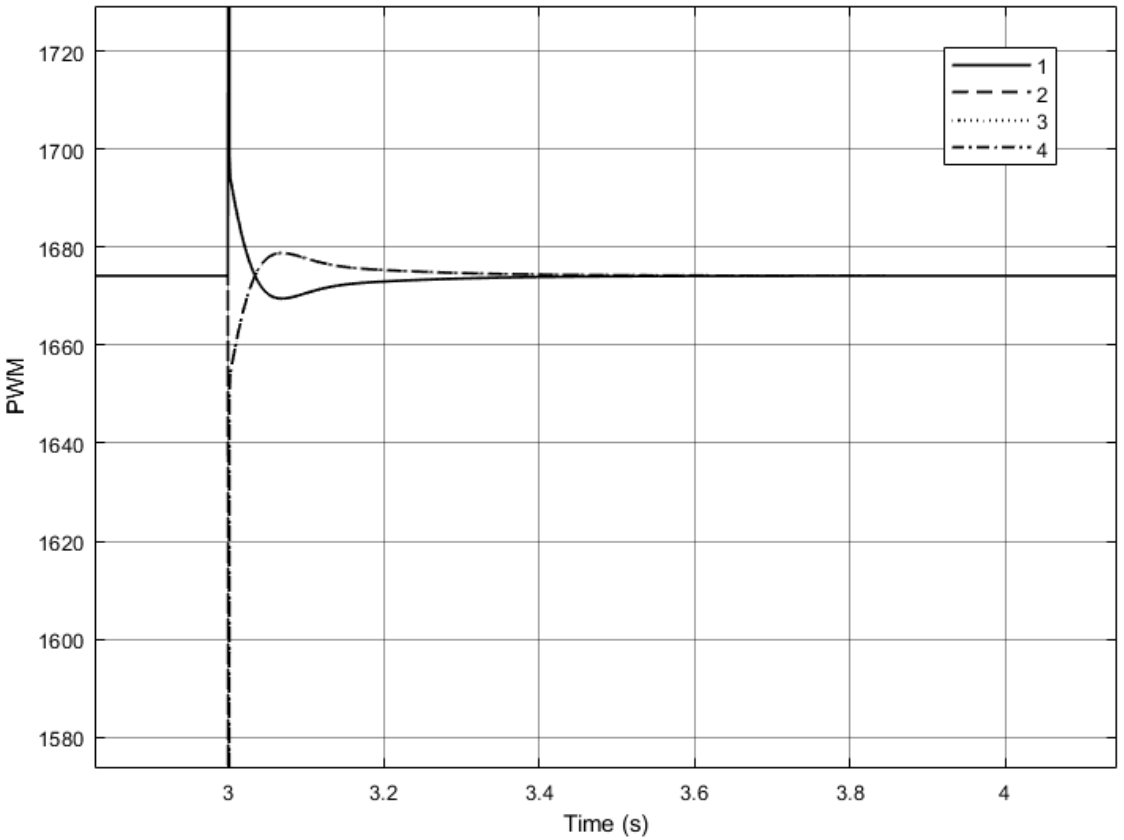


Figure. PWM values after control allocation

Basic Experiment

☐ Experimental procedure

(2) Step2: SIL simulation— Stability margin

1) Open the file

"e5\e5.1\tune\AttitudeControl_tune.slx" as shown on the right.

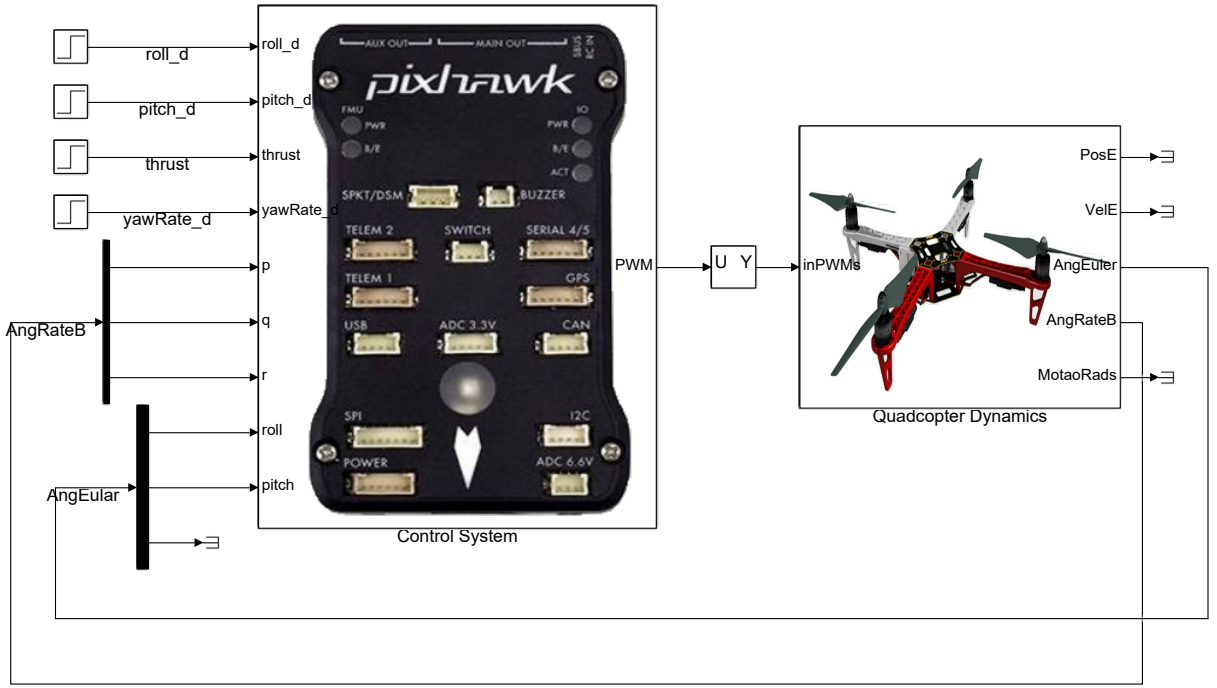


Figure. Simulink model "AttitudeControl_tune.slx"

Basic Experiment

Experimental procedure

2) Specify signals as input and output for sweeping

The module of the attitude control system can be found in the submodule “Control System”- “Attitude Control”.

There are four channels in the attitude control system,namely roll, pitch, yaw, and thrust channels.Here, the thrust channel is set to 0.6085(thrust percentage is 60.85%),which is the thrust value for hovering.

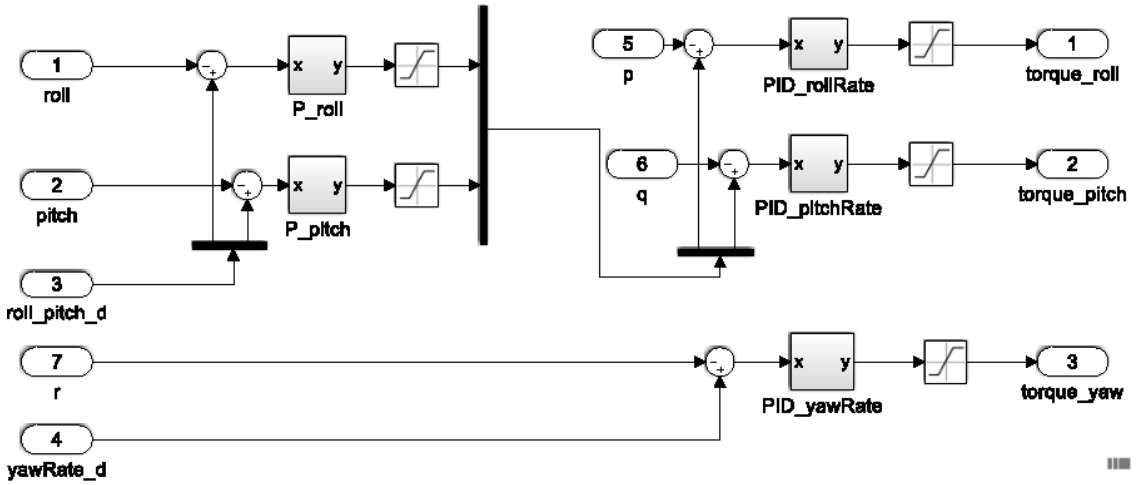


Figure. Simulink model “AttitudeControl_tune.slx”

Basic Experiment

Experimental procedure

2) Specify signals as input and output for sweeping

Choose one of the other three channels such as the pitch channel for sweeping. Specify a signal as the input using the following procedure. In the “Simulink Editor”, right-click on the signal and then select “Linear Analysis Points” - “Open-loop Input” from the context menu. Specify a signal as the output. Using the procedure followed for setting: it is the same as the input signal, but select “Open-loop Output” as shown on the right.

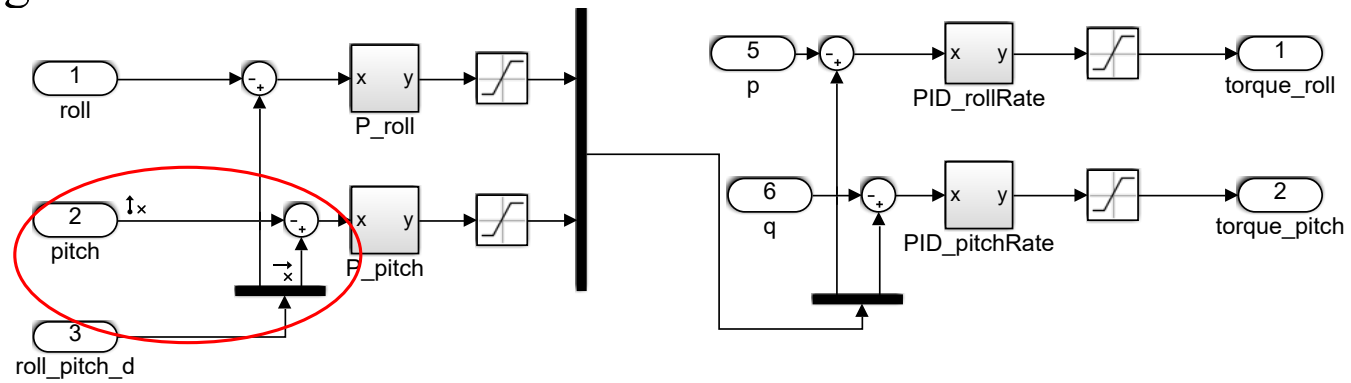


Figure. Specified Signals

Basic Experiment

☐ Experimental procedure

3) Get the Bode plot

Select “Analysis” - “Control Design” - “Linear Analysis”, then, from the context menu, select “Linear Analysis” and click “Bode” to get the Bode plot.

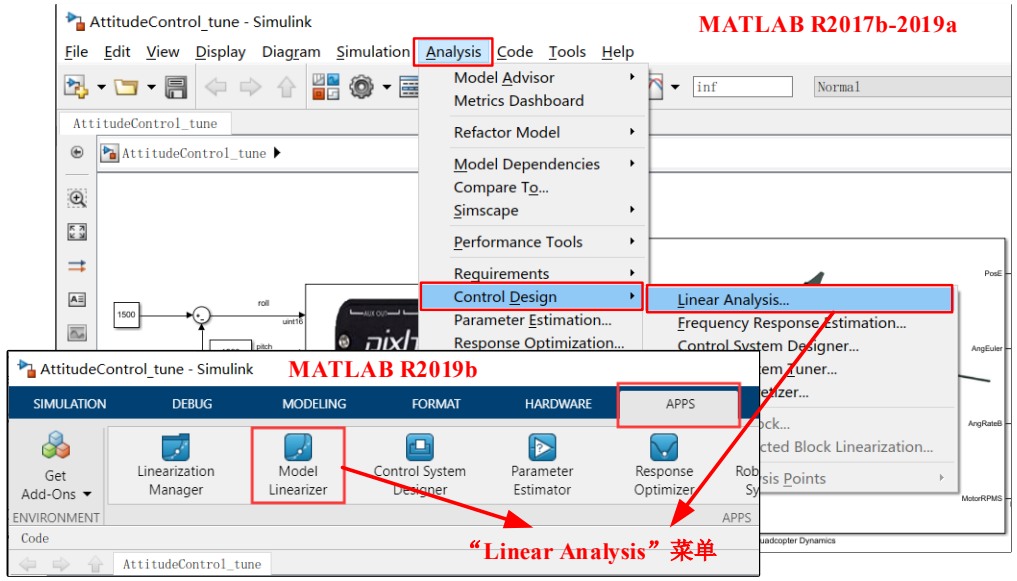


Figure. Way to open “Linear Analysis”

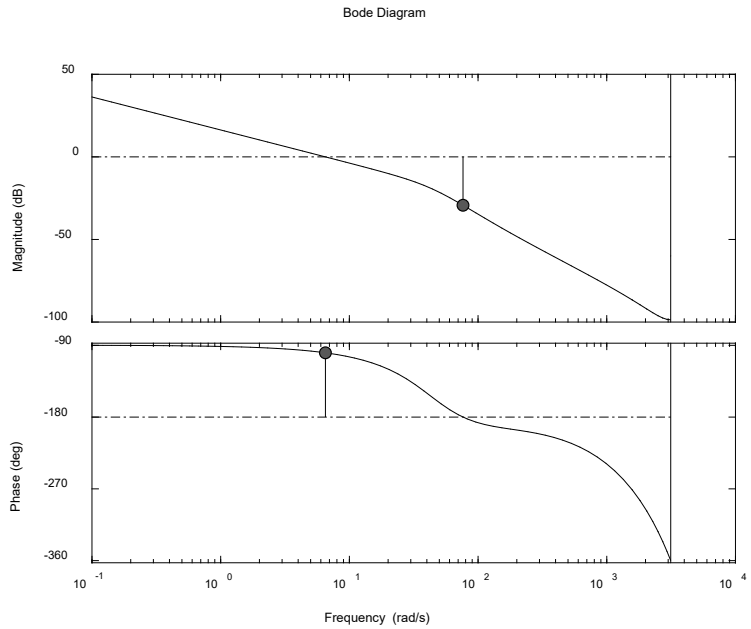


Figure. Open-loop Bode plot of pitch angle loop

Basic Experiment

☐ Experimental procedure

(3) Step3: SIL simulation— Step response

1) Specify signals as input and output to obtain a step response curve of the pitch channel

Specify a signal as the input using the following procedure. In “Simulink Editor”, right-click the signal, then select “Linear Analysis Points” - “Open-loop Input” from the context menu.

Then, specify a signal as the output same as the way like the input setting procedure expect for selecting “Output Measurement”.

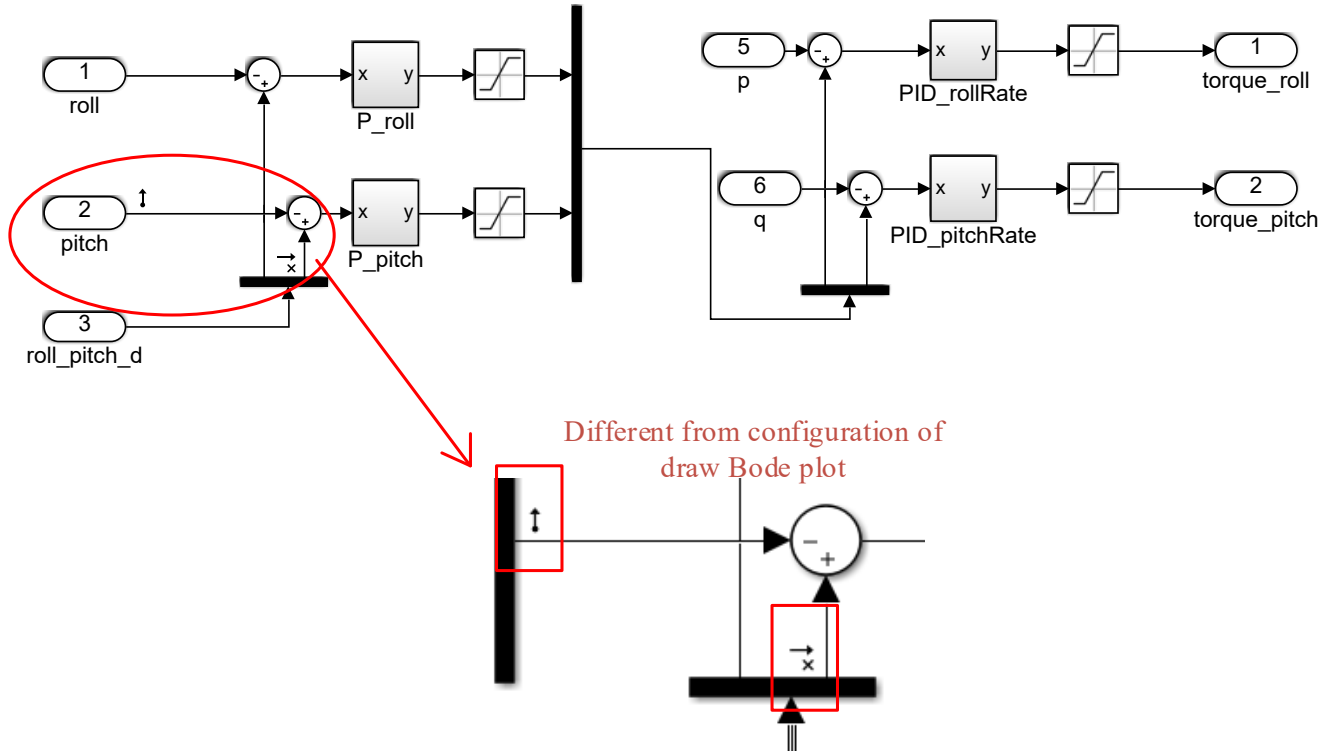


Figure. Specified signals of SIL simulation

Basic Experiment

Experimental procedure

2) Obtain the pitch step response

Go to “Linear Analysis” in Simulink and click “Step” to get the step response, as shown on the right.

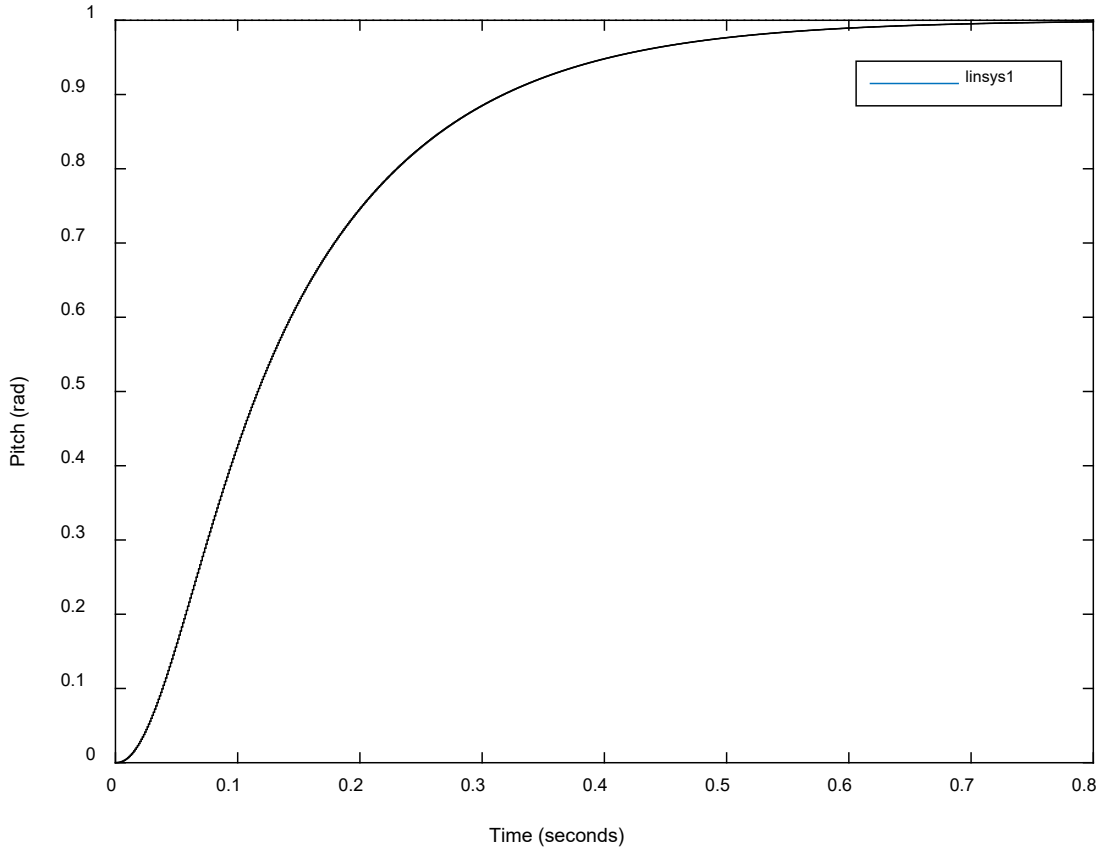


Figure. Pitch angle step response

Basic Experiment

☐ Experimental procedure

(4) Step4: HIL simulation

1) Open Simulink file for HIL

Open the Simulink file "e5\e5.1\HIL\AttitudeControl_HIL.slx", as shown on the right. It should be noted that "Control System" here is the same as that in the SIL simulation.

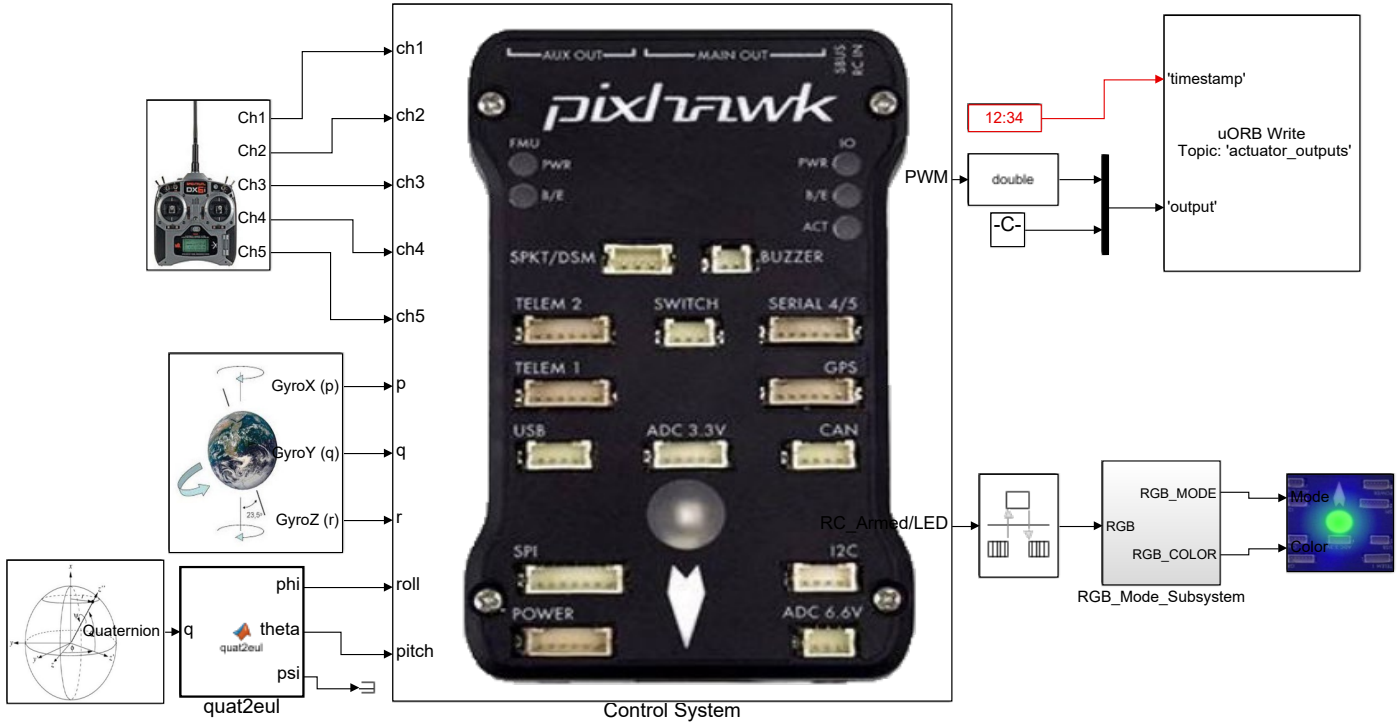


Figure. Control allocation verification model, Simulink model "AttitudeControl_HIL.slx"

Basic Experiment

☐ Experimental procedure

2) Connect hardware

It should be noted that the airframe type "HIL Quadcopter X" should be selected in HIL simulation.

Figure. Connection between Pixhawk hardware and RC receiver

Basic Experiment

☐ Experimental procedure

3) Compile and upload code

Compile the HIL simulation model and upload the file to the given Pixhawk autopilot. Later, the designed attitude control program can be run on Pixhawk autopilot.

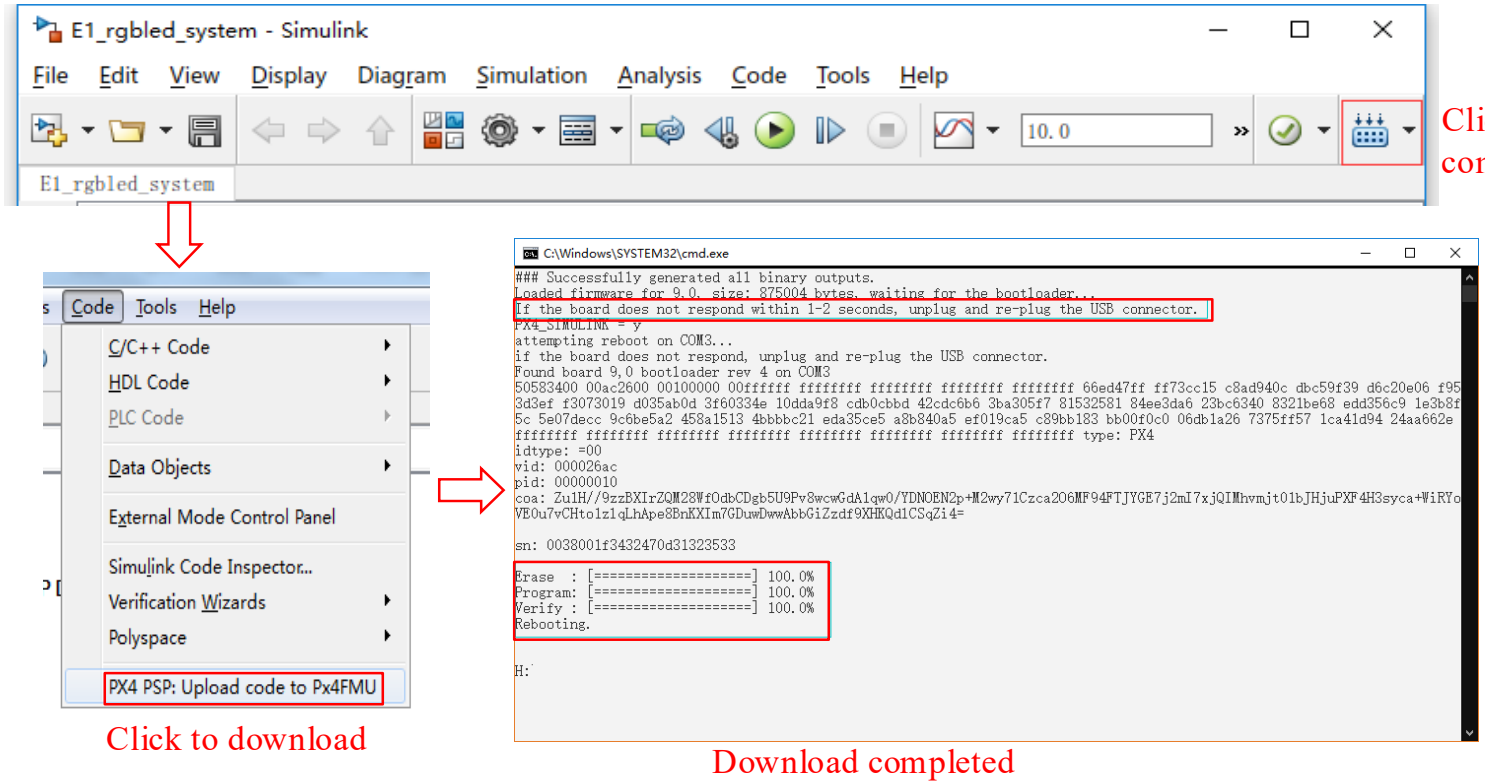


Figure. Code compilation and upload process

Basic Experiment

☐ Experimental procedure

4) Configure CopterSim

Double-click on the desktop shortcut CopterSim to open it. Readers can choose different propulsion systems using the following procedure. Click on “Model Parameters” to customize the model parameters and, then click on “Store and use the parameters” to make them available. The software will automatically match the serial port number. Readers would click the“Run”button to enter the HIL simulation mode. After that, readers could see the message returned by the Pixhawk autopilot in the lower-left corner of the interface.

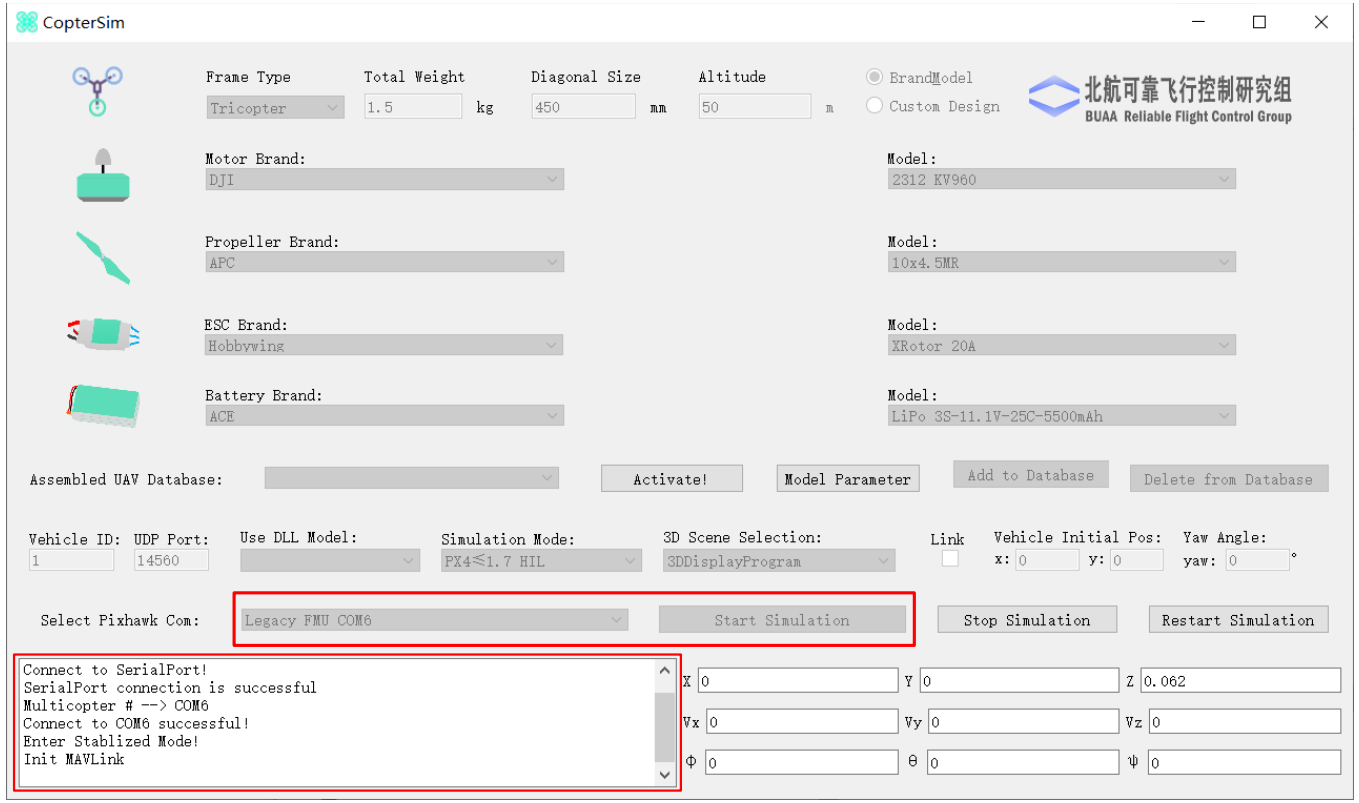


Figure. User interface of CopterSim

Basic Experiment

☐ Experimental procedure

5) Open 3DDisplay

Double-click on the desktop shortcut 3DDisply to open it.

6) Simulation performance

Arm the quadcopter for manual control using the given RC transmitter. As shown on the right. The quadcopter flight status is displayed on the left side of 3DDisplay's interface, the real-time flight data is plotted in the upper right corner, and the multicopter motion trajectory is plotted in the lower right corner.

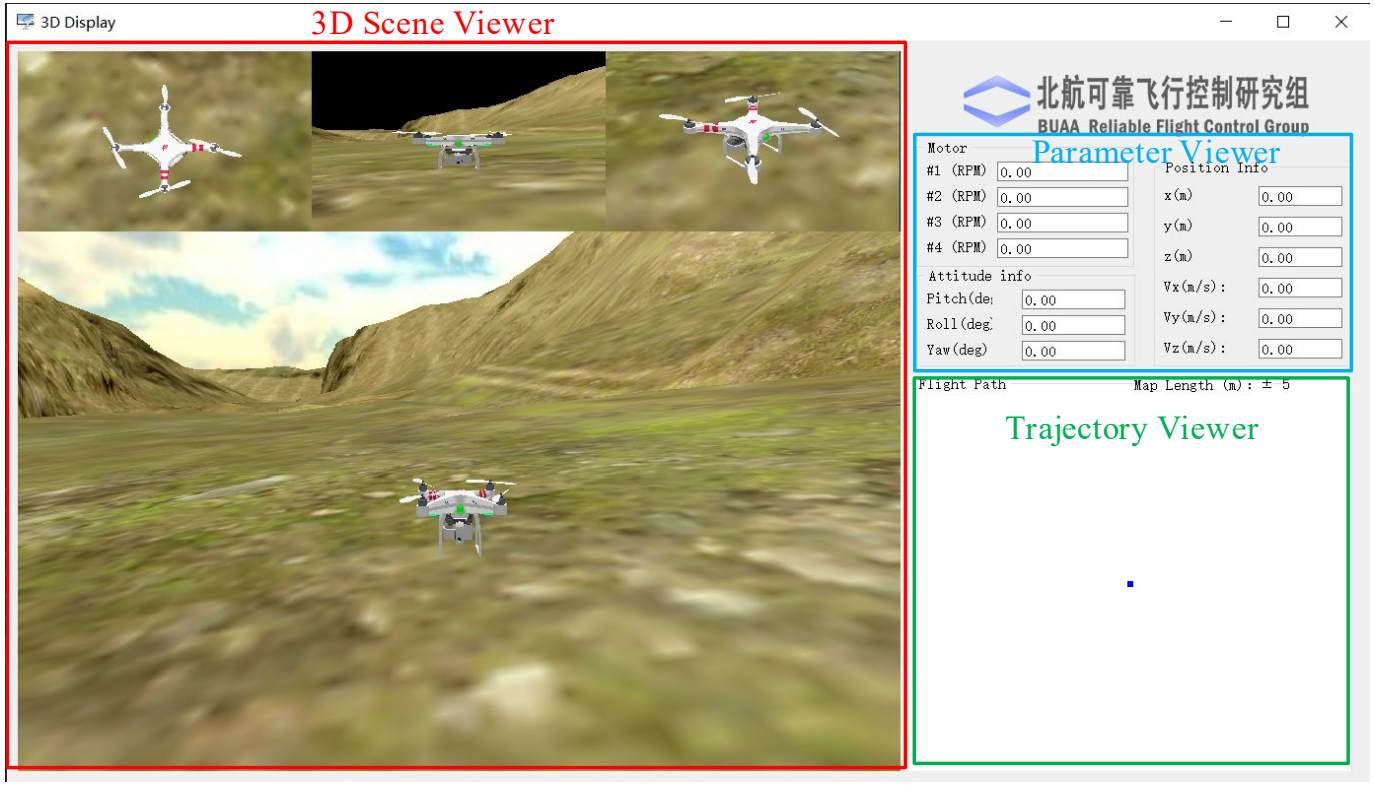


Figure. User interface of 3DDisplay

Analysis Experiment

☐ Experimental Objective

■ Things to prepare

(1) Software: MATLAB R2017b or above, Simulink-based Controller Design and Simulation Platform, Instructional Package “e5.2”(https://flyeval.com/course).

■ Objectives

(1) Adjust the PID controller parameters to improve the control performance, record the overshoot and settling time, and obtain a group of satisfied parameters;

(2) Sweep the system to draw the Bode plot based on the satisfied parameters. Observe the system amplitude versus frequency response curve and the phase versus frequency response curve. Finally, analyze the stability margin.

Analysis Experiment

Experimental Procedure

(1) Step1:Initial model setup

In this section, to adjust PID controller parameters, the Simulink-based controller design and simulation platform in the basic experiment should be simplified. The modified model is shown in the file "e5\e5.2\AttitudeControl_tune.slx".

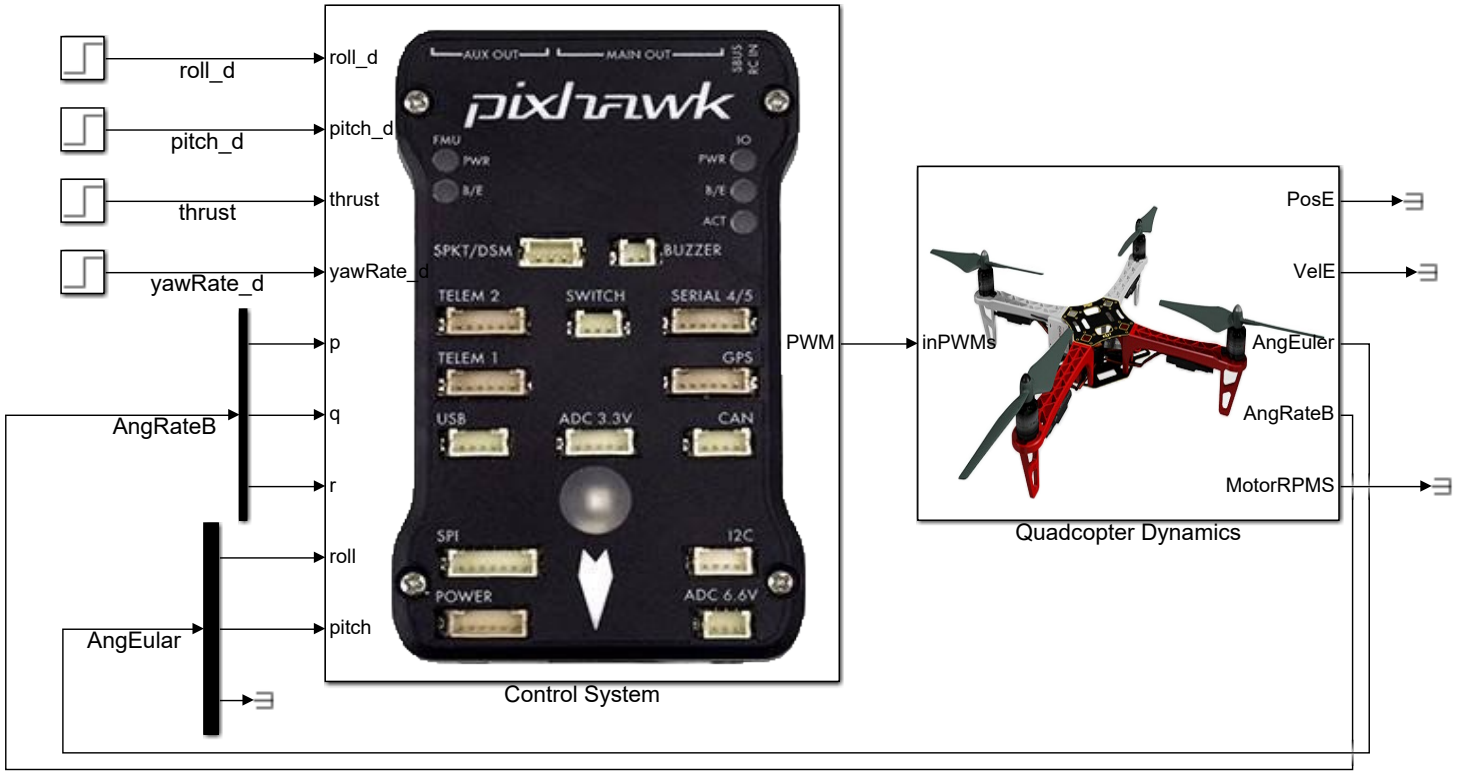


Figure. Simulink model "AttitudeControl_tune.slx"

Analysis Experiment

Experimental Procedure

(1) Step1:Initial model setup

As the quadcopter is symmetrical, the PID parameters for the roll angle and pitch angle are the same. Hence, only one group of the roll or pitch angle parameters needs to be adjusted. The adjustment of PID parameters of the pitch angle is described below. First, let the quadcopter hover at an initial altitude of 100m by setting the thrust value to 0.6085 (thrust percentage is 60.85 %) and the initial speed of the motor to 557.1420rad/s. Modify the corresponding parameters in the file “Init_control.m”.

```
ModelInit_PosE=[0,0,-100];
ModelInit_VelB=[0,0,0];
ModelInit_AngEuler=[0,0,0];
ModelInit_RateB=[0,0,0];
ModelInit_RPM=557.1420;
```

Analysis Experiment

☐ Experimental Procedure

(2) Step2: Adjust the PID parameters for the angular velocity control loop

In the file "AttitudeControl_tune.slx", replace the desired angular velocity with the step input and left-click on "q" signal line(corresponding to the angular velocity) and step input line, select "Enable Data Logging" to get the step response.

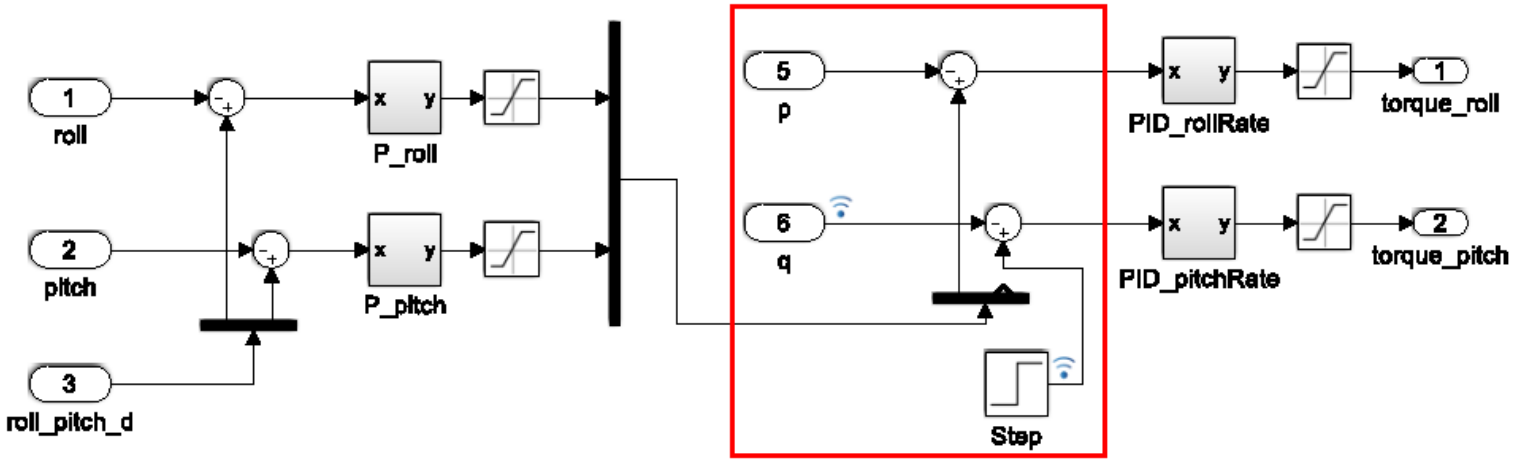


Figure. Setting step response of angel velocity control loop

Analysis Experiment

☐ Experimental Procedure

(2) Step2: Adjust the PID parameters for the angular velocity control loop

First, adjust the proportional term parameter and set the integral and derivative term parameters to 0. Later, run the file “Init_control.m” (this file must be run every time to update any altered parameters). Click on Simulink’s “Run” button to view the input and output in the “Simulation Data Inspector”. The proportional term parameter is gradually increased from a small value to a large value, which corresponds to increase “Kp_PITCH_AngleRate” variable in the file “Init_control.m”.

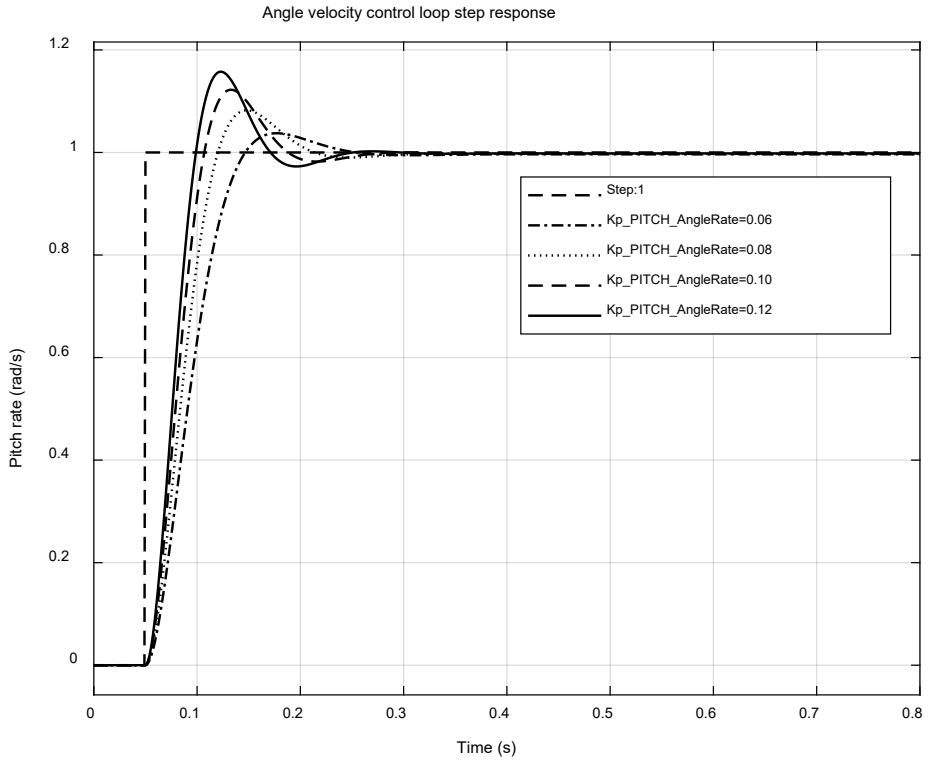


Figure. Step response with different proportional term parameters

Analysis Experiment

☐ Experimental Procedure

(2) Step2: Adjust the PID parameters for the angular velocity control loop

Next, adjust the integral and derivative term parameters, i.e., “Ki_PITCH_AgngleRate”, “Kd_PITCH_AgngleRate” variables in the file “Init_control.m”. Finally, fine tune the proportional term parameter with theresulting response shown on the right.

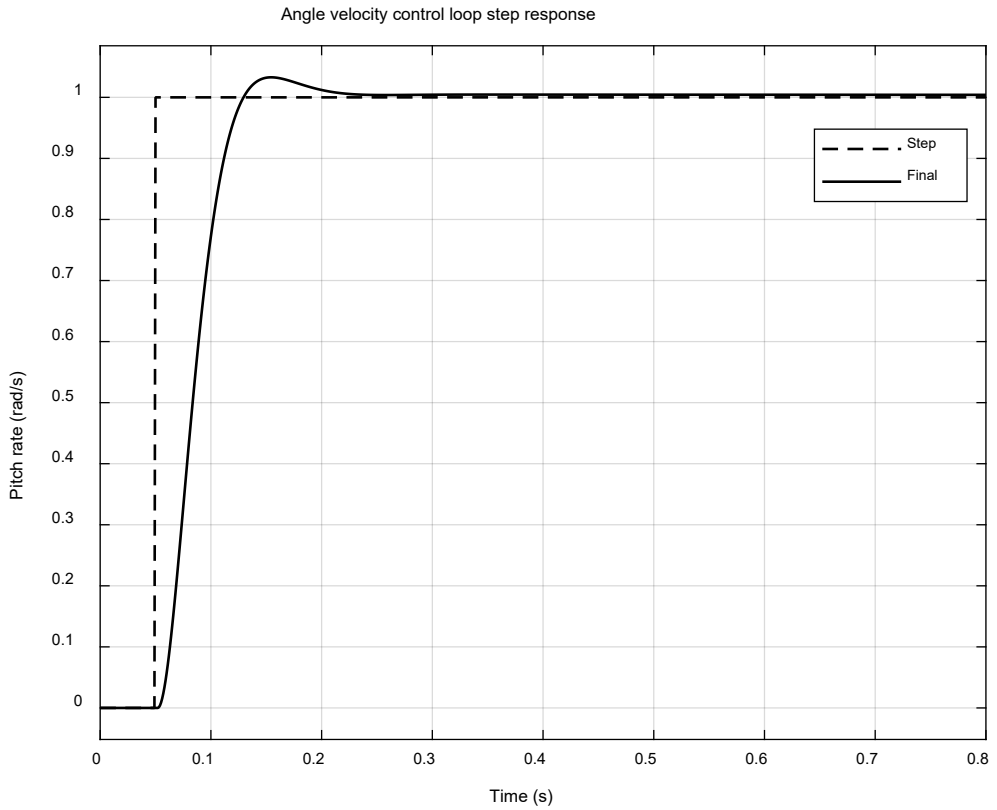


Figure. Step response with a group of satisfied PID parameters

Analysis Experiment

☐ Experimental Procedure

(3) Step3: Adjust the PID parameters for the angle control loop

Adjust the proportional term for the angle control loop, corresponding to the variable "Kp_PITCH_ANGLE" in the file "Init_control.m". Based on the angular velocity control loop parameters obtained in Step 2, replace the desired pitch angle with a step input and set the "pitch" signal line and the step input as "Enable Data Logging".

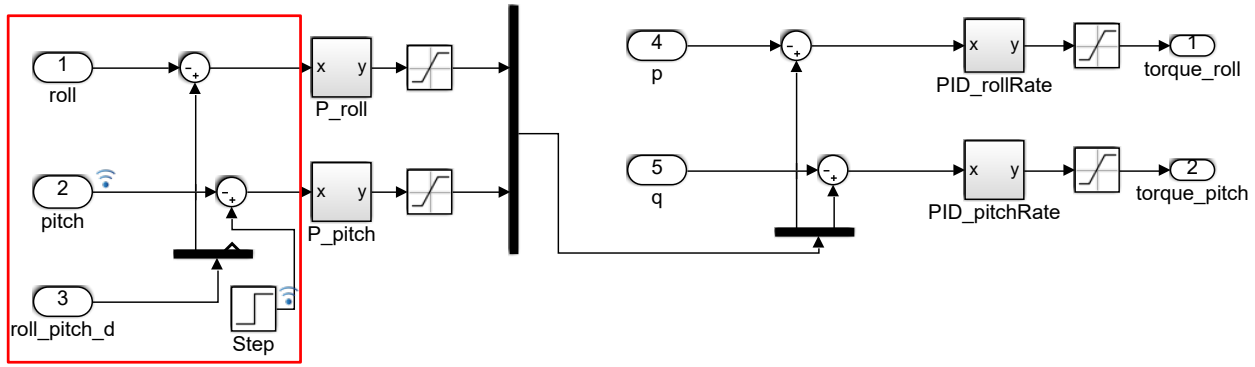


Figure. Setting step response of angle control loop

Analysis Experiment

☐ Experimental Procedure

(3) Step3: Adjust the PID parameters for the angle control loop

Increase the proportional term parameter gradually and observe the step response in the “Simulation Data Inspector”.

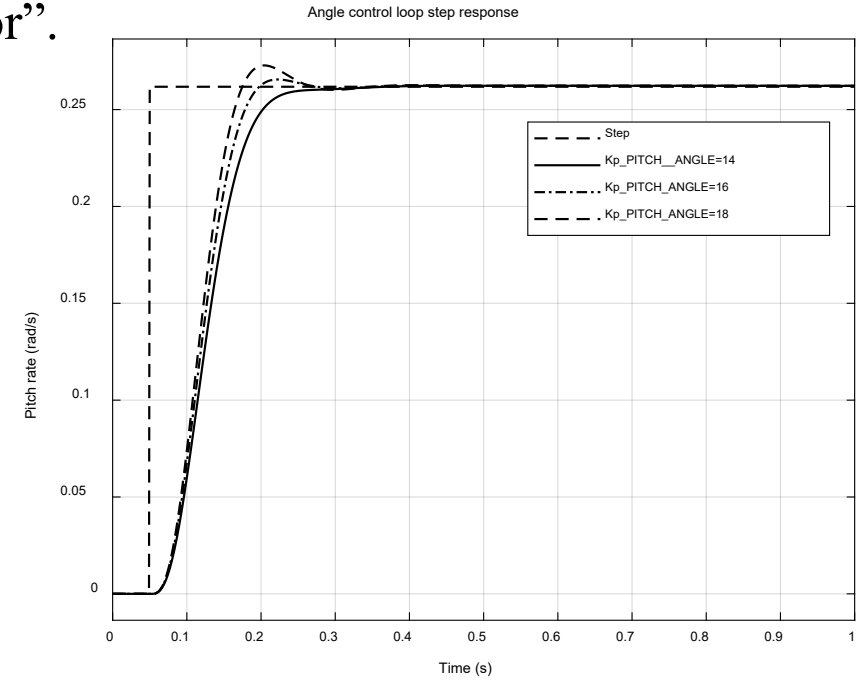


Figure. Pitch angle step response with different parameters

Kp_RP_ANGLE =14;

Kp_RP_AgngleRate = 0.10;

Ki_RP_AgngleRate = 0.02;

Kd_RP_AgngleRate = 0.001;

Analysis Experiment

☐ Experimental Procedure

(4) Step4: Sweep to get the Bode plot

Specify signals as the input and the output for Bode plot. Specify the desired pitch angle input line as “Open-loop Input”, and the actual output as “Open-loop Output”. The Bode plot obtained using these values is shown on the right.

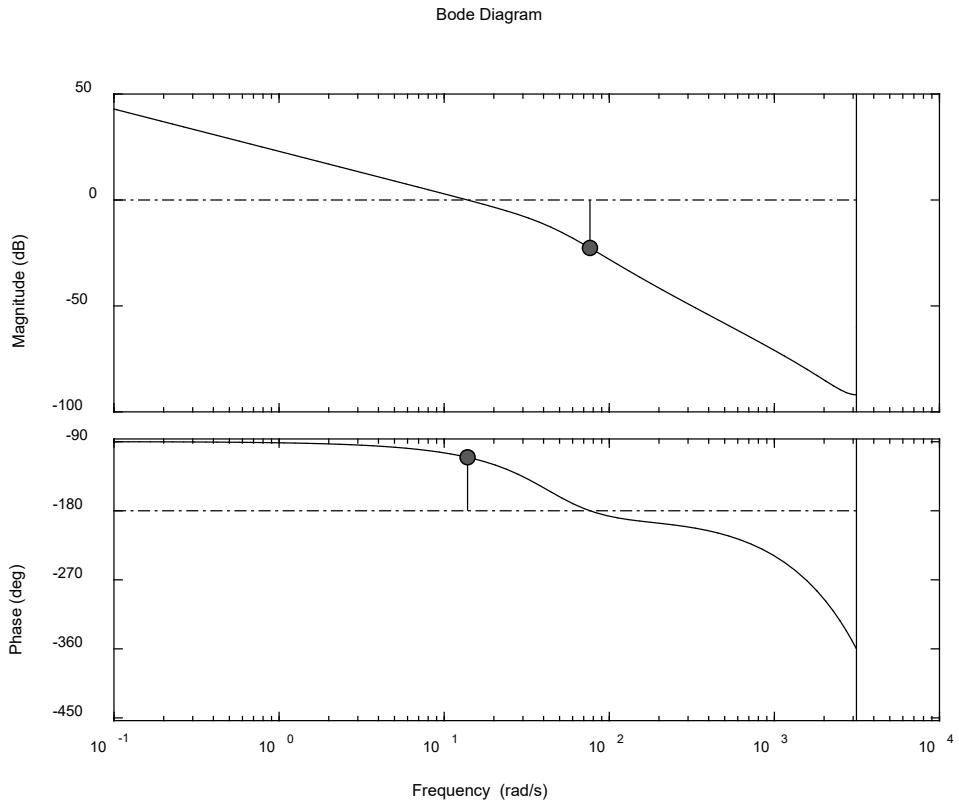


Figure. Open-loop Bode plot of pitch angle control loop

Design Experiment

Experimental Objective

■ Things to prepare

(1) Hardware: Multicopter Hardware System, Pixhawk Autopilot System;

(2) Software: MATLAB R2017b or above, Simulink-based Controller Design and Simulation Platform, HIL Simulation Platform, Instructional Package “e5.3”(https://rflysim.com/course).

■ Objectives

(1) Obtain the transfer function model of the attitude control loop, and then design a compensator for the attitude angular velocity control loop satisfying the following conditions: steady-state error $e_{rss} \leq 0.01$, phase margin > 65 and cut-off frequency > 10rad/ s. The attitude angle control loop is satisfied when cut-off frequency > 5rad/s, phase margin > 60;

(2) Complete the SIL , HIL simulation and flight test experiments with the designed controller.

Design Experiment

Experimental Design

(1) Step1: Simplify the overall structure

Take the pitch channel for example. The simplified model is shown below.

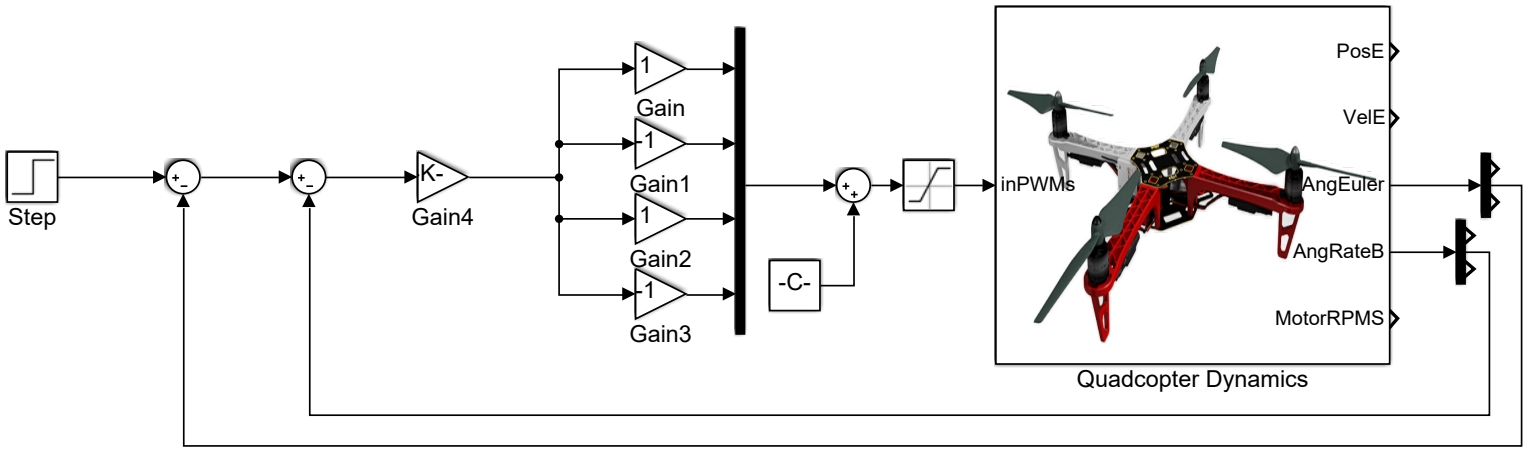


Figure. Simplified model of pitch angle control loop, Simulink model “AttitudeControl_tune.slx”

Design Experiment

Experimental Design

(2) Step2: Angle velocity-control loop analysis

The input and output are the desired angular velocity and the angular velocity, respectively. Specify these signals as the input and output for the Bode plot, as shown on the right.

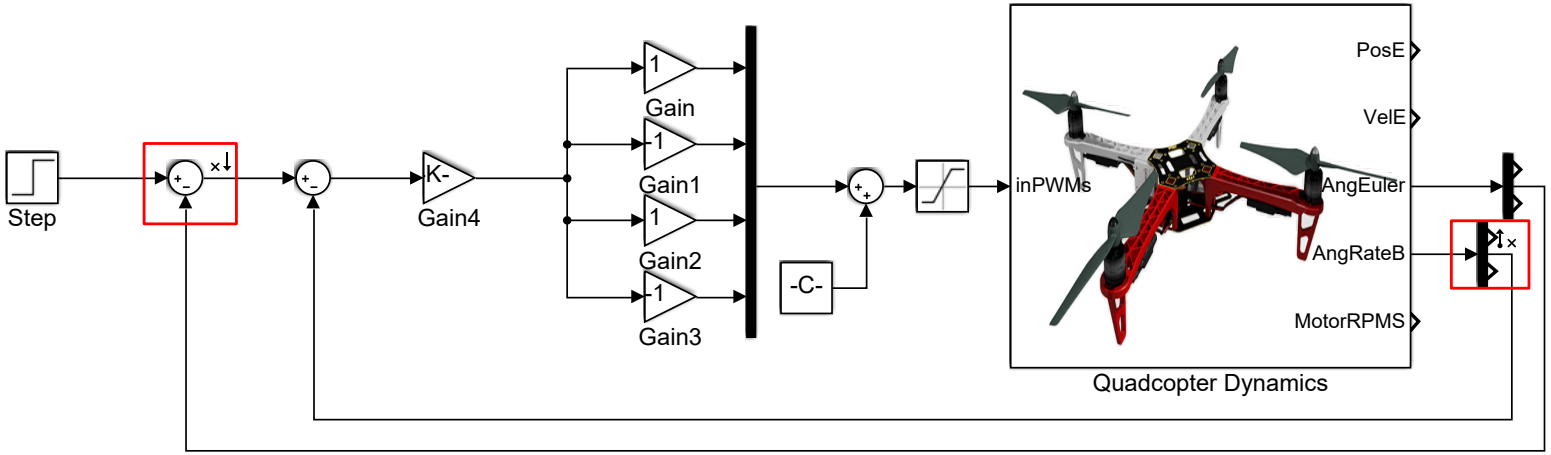


Figure. Specifying signals as input and output

Design Experiment

☐ Experimental Design

(3) Step3: Obtain the transfer function model

After the Bode plot is obtained, a variable, "linsys1" appears, in the "Linear AnalysisWorkspace". The transfer function model can be obtained using the operation shown on the right.

$$\frac{20538}{(s+9.132e-07)(s+50)} \Rightarrow \frac{410.76}{s(0.02s+1)}$$

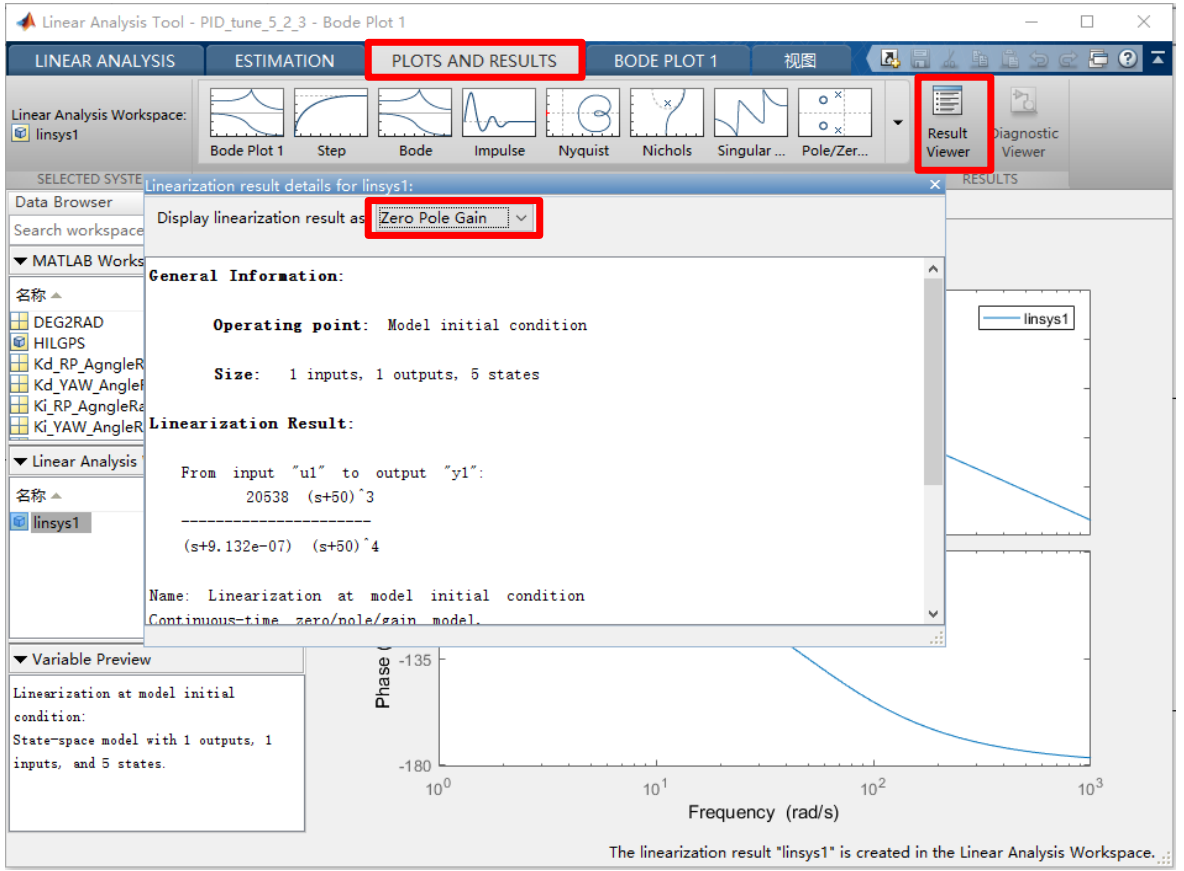


Figure. Transfer function of angular velocity control loop

Experimental Design

(4) Step4: Adjust open-loop gain

First, the open-loop gain is adjusted according to the steady-state error. When there is no compensator, the steady-state error of the system e_{rss} with an input $r(t)=t$ can be obtained by the final value theorem as

$$e_{rss}=\frac{1}{K}$$

where K is the open-loop gain. Because $e_{rss}\le 0.01$,$K\ge 100$, no adjustment is needed because its open-loop $K=410.76$ ·

Design Experiment

(5) Step5: Design a compensator for the angular velocity control loop

First, draw a Bode plot of the initial system, as shown on the right.

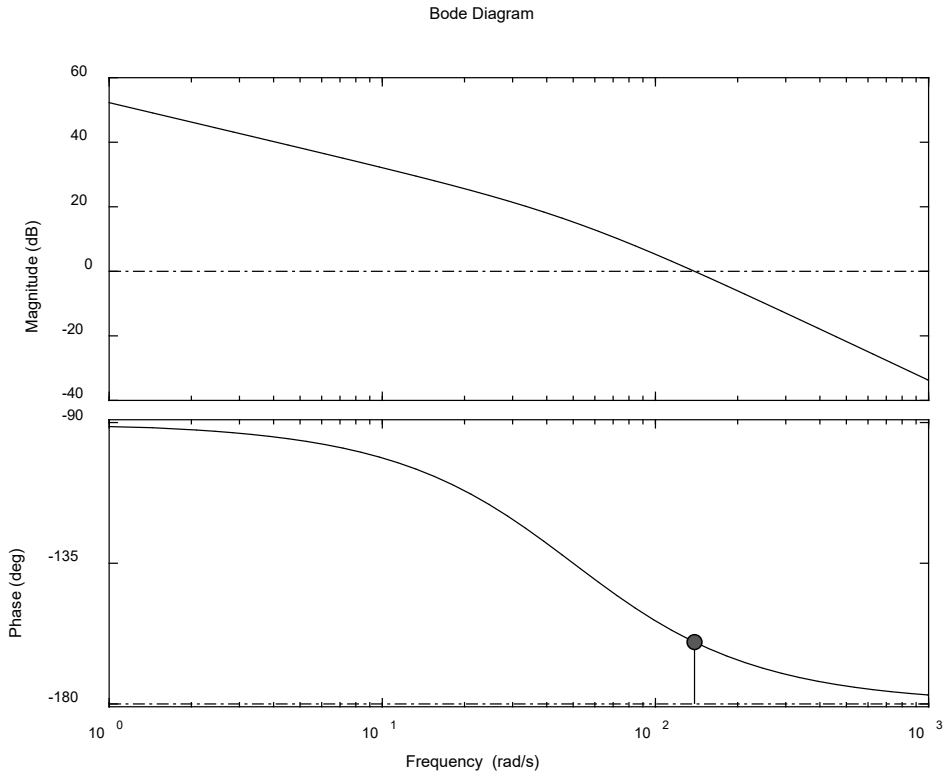


Figure. Angle velocity control open-loop Bode plot without compensation

Design Experiment

(5) Step5: Design a compensator for the angular velocity control loop

In the Bode plot, it can be observed that the phase margin $\gamma = 19.8^{\circ}$ does not meet the requirements. Consider designing a phase-lag compensator as it not only ensures that the phase margin meets the specified requirements but also improves the system's ability to suppress high-frequency noise. Consider setting the cut-off frequency at $\omega_c^{'} = 12.9 rad/s$, where the phase margin is 76° which satisfies the requirements, the gain margin is 29.8dB. At a frequency $\omega_c^{'}$, the corresponding amplitude should be 0dB after compensation. Based on the typical lag-lead compensation Bode plot, we can obtain the following relationship.

$$20\lg b + 20\lg\left|G(j\omega_c')\right| = 0$$

Thus, b=0.0324. To reduce the phase lag effect of the compensator on the curve at frequency $\omega_c^{'}$, the cut-off frequency of the compensator $(bT)^{-1}$ should be located ten decades away from $\omega_c^{'}$, i.e.,

$$(bT)^{-1} = 0.1\omega_c'$$

Hence, $T = 23.9558$.The transfer function of the compensator is given as follows.

$$G_c(s) = \frac{1+bTs}{1+Ts} = \frac{1+0.775194s}{1+23.955779s}$$

Design Experiment

☐ Experimental Design

(5) Step5: Design a compensator for the angular velocity control loop

The Bode plot obtained after adding the compensator is shown below.

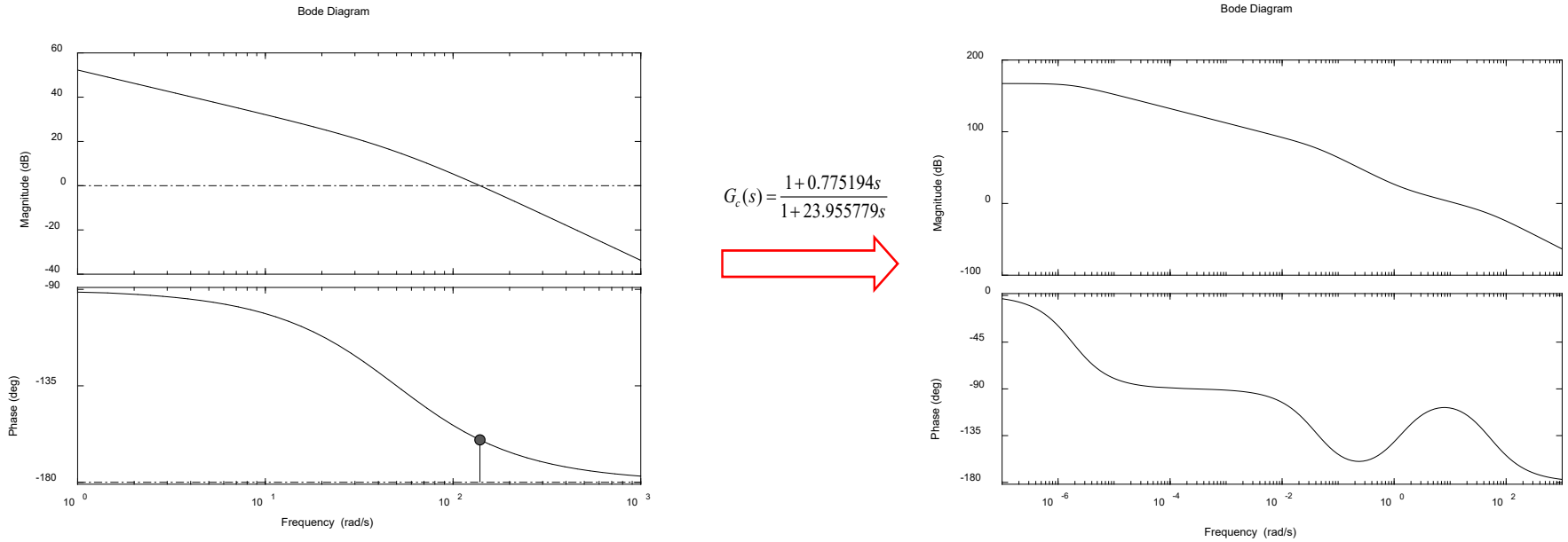


Figure. Open-loop Bode plot of angle velocity control loop before and after compensation

Design Experiment

☐ Experimental Design

(6) Step6: Design a compensator for the angle control loop

The angle control loop is also designed as a feedback closed loop, based on which, a compensator is designed. Specify signals as the input and the output, as shown below.

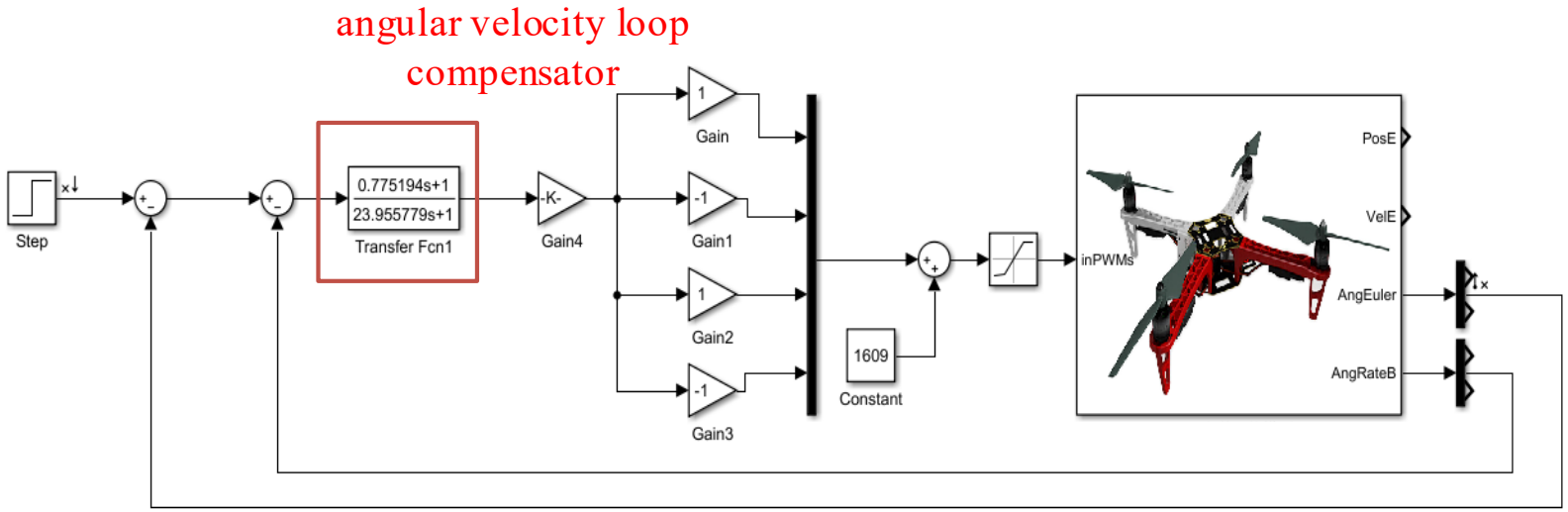


Figure. Uncompensated model of the pitch angle control loop, Simulink model "AttitudeControl_tune.slx"

Design Experiment

☐ Experimental Design

(6) Step6: Design a compensator for the angle control loop

The resultant open-loop Bode plot is obtained shown on the right.

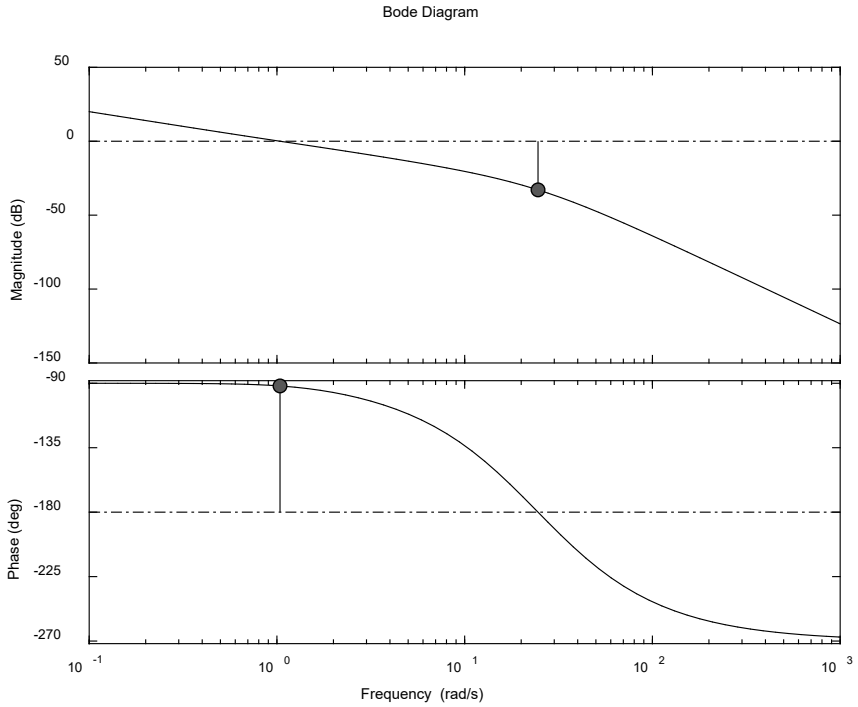


Figure. Open-loop Bode plot of angle control loop without compensation

Experimental Design

(6) Step6: Design a compensator for the angle control loop

It can be observed that the cut-off frequency is 1.04 rad/s at a phase margin of 88.1. According to the design requirements of the attitude angle control loop, consider increasing the open-loop gain. According to the figure, we hope to locate the new cut-off frequency at ω=5.3rad/s, where the corresponding amplitude is -14dB. This implies that after adding the compensator, the amplitude of Bode plot at the frequency ω=5.3rad/s needs to be 0dB. Then,

$$20\lg\left(K_2\right)=14$$

Thus, $K_2=5.0119$.

Design Experiment

(6) Step6: Design a compensator for the angle control loop

The Bode plot obtained after adding the compensator, where the cut-off frequency is 5.3 rad/s and the amplitude margin is 67.1°.

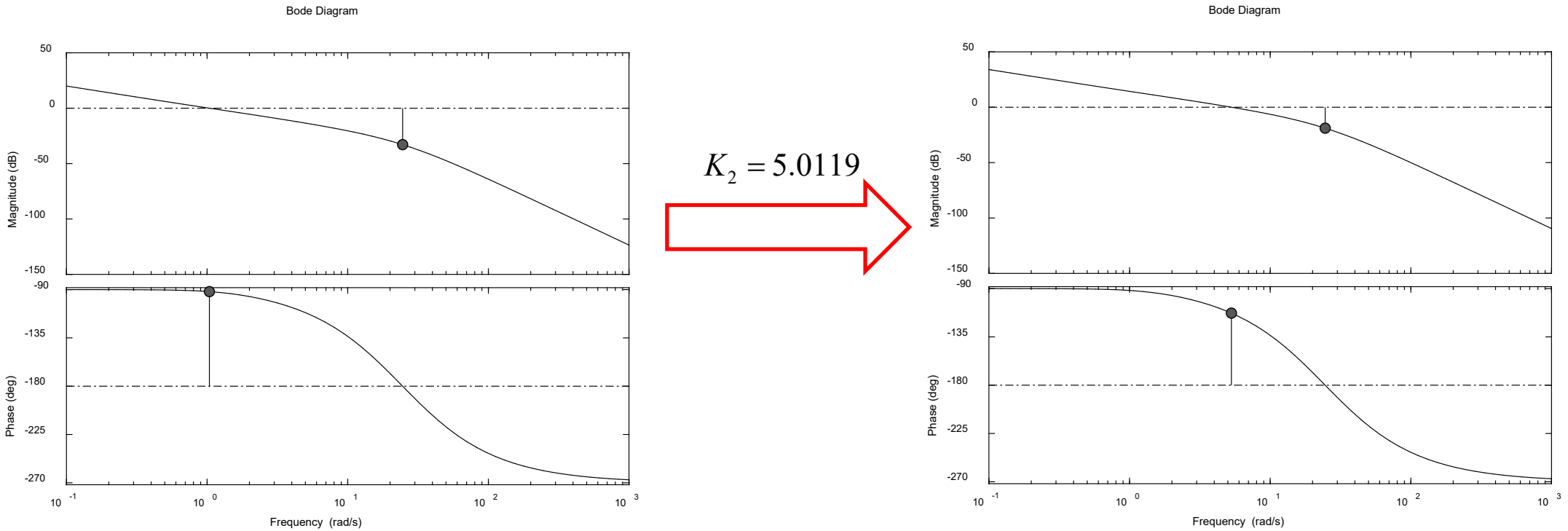


Figure. Open-loop Bode plot of angle control loop after compensation

Design Experiment

☐ Simulation procedure

(1) Step1: Discretize the continuous-time compensator

The designed compensator is an s transfer function, which has to be discretized so that it can be run on the Pixhawk autopilot, a digital computer. The “c2d” function in MATLAB is used as:

H = tf([num], [den])

Hd = c2d(H, Ts, 'foh')

Here, “num” is the transfer function numerator coefficient vector, “den” is the transfer function denominator coefficient vector, and “ Ts” is the sample time, “Ts= 0.004s”.

The s transfer function is converted into a z transfer function as follows

$$G_c(s)=\frac{1+0.775194s}{1+23.955779s} \rightarrow G_c(z)=\frac{0.03236\ z - 0.03219}{z - 0.9998}$$

Design Experiment

☐ Simulation procedure

(2) Step2: Replace the control model

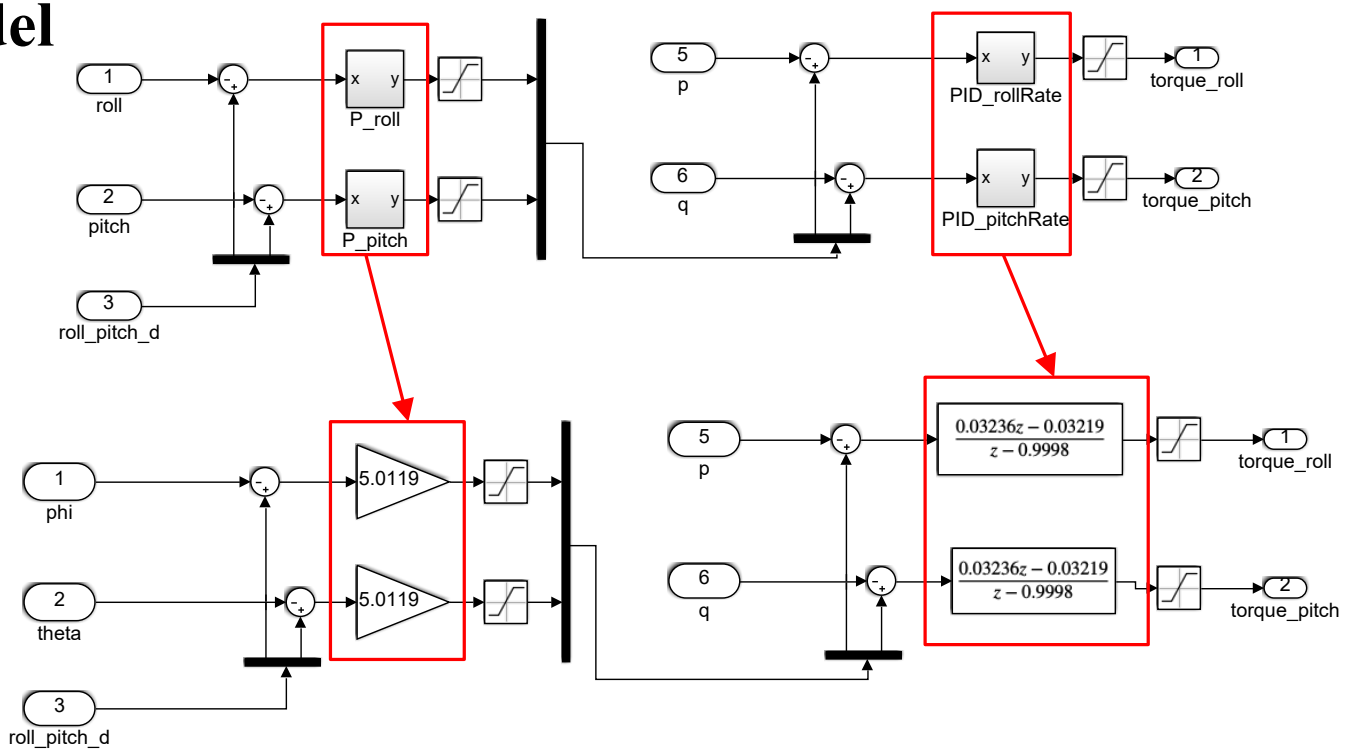


Figure. PID controller discretized for the HIL simulation model

Design Experiment

☐ Simulation procedure

(3) Step3: HIL simulation

It is observed that by releasing of the control stick, the quadcopter can quickly return to its previous stable state without significant oscillations.

Figure. Arm the quadcopter

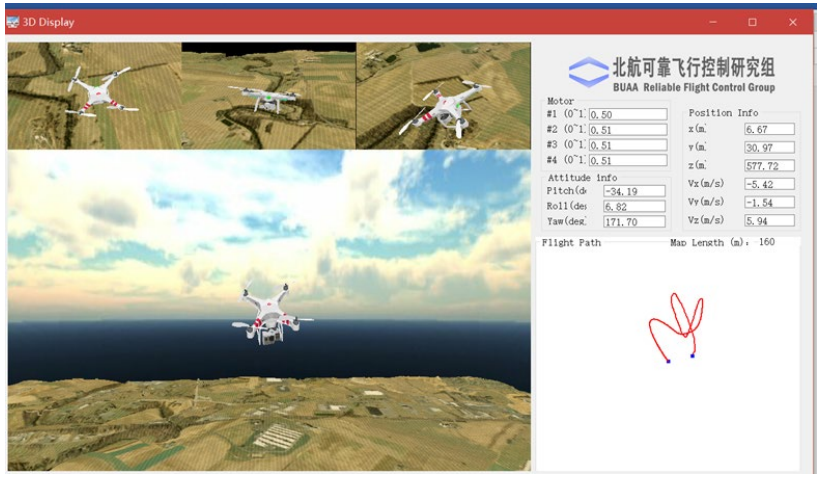


Figure. Move the control stick to change pitch angle

Design Experiment

☐ Flight Test Procedure

(1) Step1: Quadcopter configuration

The multicopter used in the outdoor flight tests is an F450 quadcopter For outdoor flight tests, the airframe of Pixhawk should be changed from “HIL Quadcopter X” to “DJI Flame Wheel F450” in QGC and all sensors should also be calibrated in QGC.

Figure. F450 airframe schematic

Design Experiment

☐ Flight Test Procedure

(2) Step2: Simulink model for flight test

Compared with the model in the HIL experiment, the flight test model is changed the PWM output. A new data recording module is added to the model, A “invalid.msg.specified” warning block appears automatically when the Simulink model is opened.

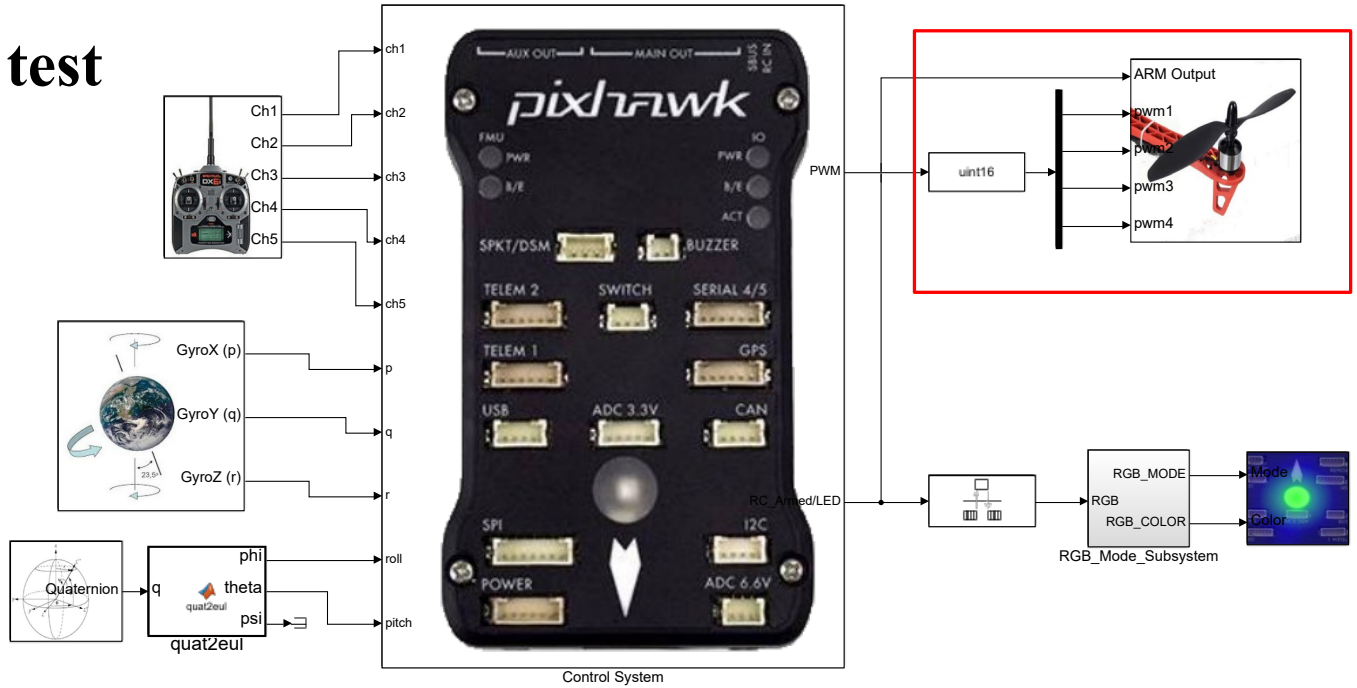


Figure. Model for flight test, Simulink model ”AttitudeControl_FLY.slx”

Design Experiment

☐ Flight Test Procedure

Modify PX4 Firmware to enable customized log file.

(2) Step2: Simulink model for flight test

1) Customize the message file. Create customized message.“costom_attctrl_e5.msg” is the customized message, which should be put in “Firmware/msg/”. Please refer to the demo file in “e5\e5.4\PSPfile”. Note that, for PX4 firmware 1.9 and Above, the “uint64 timestamp” should be added on the top of the msg file.

2) Modify “Firmware/msg/CmakeLists.txt” . Add the follow words below the line “set(msg_files” :

costom_attctrl_e5.msg

3) Modify “Firmware/src/modules/logger/logger.cpp” (If you are using PX4 1.11 and above, please modify “Firmware/src/modules/logger/logged_topics.cpp”) instead. Add the follow words in add_default_topics():

add_topic("costom_attctrl_e5", 4);

where “costom_attctrl_e5” is the name of message, “4” is logging period. That is, the system records the data with a sampling period of 4ms (i.e., 250Hz).

```
1  #attitude data
2  float32[2] euler_rp
3  float32 yawrate
4  #desired attitude data
5  float32[2] euler_rp_d
6  float32 yawrate_d
7
```

Design Experiment

☐ Flight Test Procedure

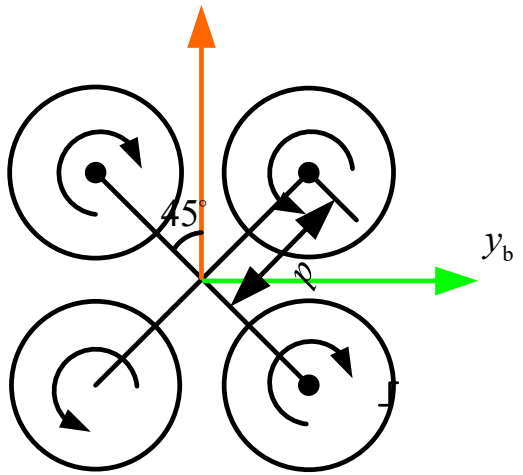
(3) Step3: Upload code

This process is similar to that used for compiling and uploading the code in HIL simulation.

(4) Step4: No propellers test

1) Thrust channel

- Remove the propellers and, then connect the battery supply
- Turn on the RC and switch the RC CH5 stick to arm the quadcopter
- Pull up the left-hand stick on the RC transmitter (CH3) and observe if the motors rotate correctly



2) Roll channel

- Move the right-hand stick on the RC transmitter (CH1) to the left
- Observe whether the speeds of the motors #1 and #4 increase or the speeds of the motors #2 and #3 decrease

Design Experiment

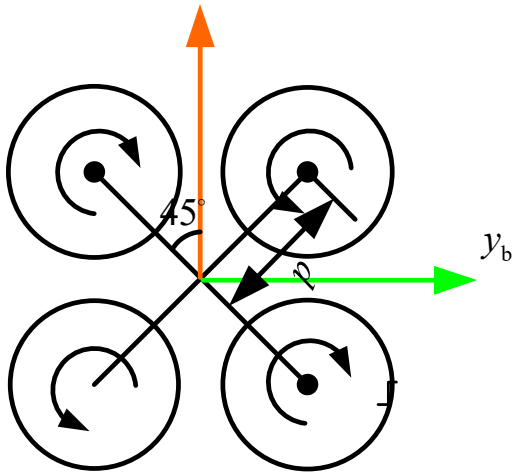
Flight Test Procedure

3) Pitch channel

- Pull up the right-hand stick on the RC transmitter(CH2)
- Observe whether the speeds of motors #2 and #4 increase or the speeds of motors #1 and #3 decrease.

4) Yaw channel

- Move the left-hand stick on the RC transmitter (CH4) to the right.
- Observe whether the speeds of motors #1 and #2 increase or if the speeds of motors #3 and #4 decrease.

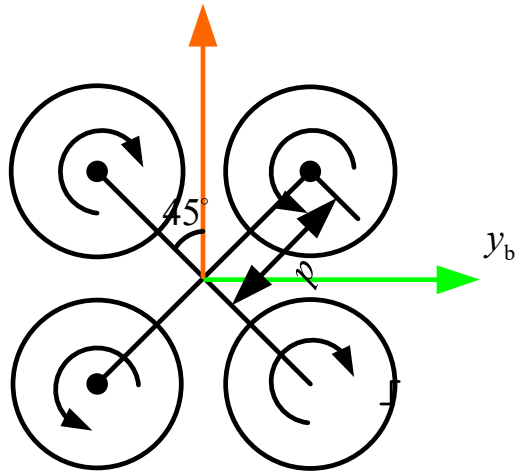


Design Experiment

Flight Test Procedure

(4) Step4: No propellers test

5) Automatic control test



- Keep the CH3 channel control stick in the middle and hold the quadcopter horizontally. In next step, tilt the quadcopter to the left. Observe whether the speeds of motors #2 and #3 increase or the speeds of motors #1 and #4 decrease.
- Keep the quadcopter horizontal. Tilt it forward. Observe whether the speeds of motors #1 and #3 increase or the speeds of motors #2 and #4 decrease.
- Keep the quadcopter horizontal. Yaw the quadcopter to the right and observe if the speeds of the motors #1 and #2 decrease or the speeds of motors #3 and #4 increase.

If the quadcopter performs well in all the steps , then it is considered that the flight controller can accurately control the attitude of quadcopter.

Design Experiment

☐ Flight Test Procedure

(5) Step5: Indoor stand test

Install the quadcopter on a test stand and install the propeller as shown on the right. Arm the quadcopter and test the flight.

Figure. A quadcopter on a test stand

Design Experiment

☐ Flight Test Procedure

(6) Step6: Outdoor flight test

To ensure safety, a rope is tethered to the quadcopter, and the other end is tethered to a heavy object. The remote pilot maintains a safe distance from the quadcopter during flight. Notice that it may be difficult to control the altitude as there is no position feedback. Do not rotate the thrust control stick too fast. Furthermore, it is better to let the stick be near the middle position, and rotate the stick smoothly.

Figure. Safety precautions for outdoor flight test

Design Experiment

☐ Flight Test Procedure

(7) Step7. Analyze the data

1) Take out the SD card in Pixhawd, read out the logger file “log001.ulg”, and then copy the file to “e5\e5.4\”;

2) Use function “ulog2csv” to convert “.ulg” file to “.csv” file for analysis. For example, run the function “ulog2csv(’log001.ulg’,’log001’)” can extract the contents of the file “log001.ulg” to the folder “log001”. And the customized message data is in the file “log001_costom_attctrl_e5_0.csv”. The stand test data is shown on the right.

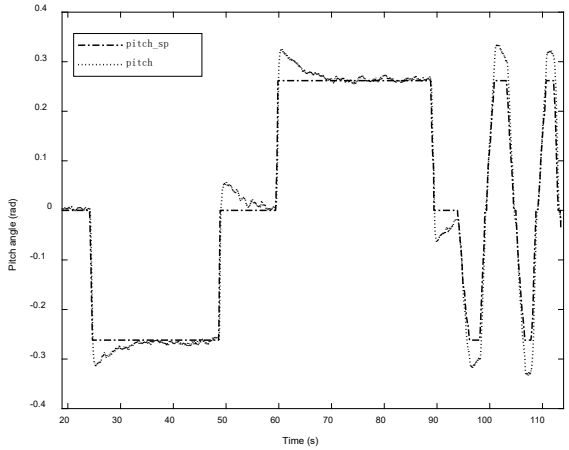


Figure. Pitch step response of a quadcopter on a test stand

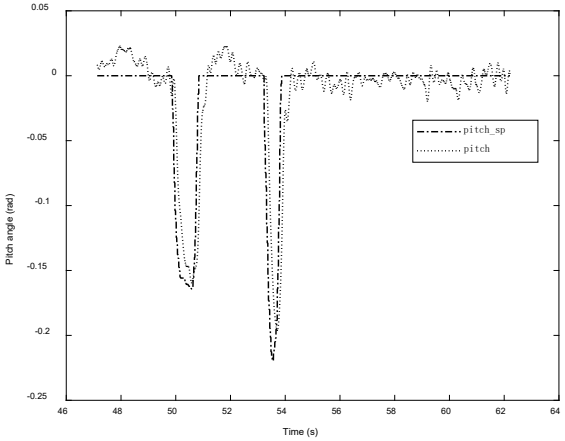


Figure. Pitch step response of outdoor flight test

Summary

(1) Based on the attitude model of the quadcopter, a widely-used PID control method is used, and the design of the attitude controller is completed in Simulink and MATLAB. The simulation performance is displayed in FlightGear.

(2) Use the PSP tool of Simulink to generate the embedded code and upload it to the Pixhawk autopilot for HIL simulation.

(3) Adjust the parameters of the PID controller, try to obtain the satisfied parameters, and use the “System Analysis Tool” in MATLAB to obtain the open-loop Bode plot of the entire system to observe the phase margin and gain margin of the corresponding closed-loop system.

(4) Lead-and-lag compensators are designed for both the angle control loop and angular velocitycontrol loop. Furthermore, the design is verified by HIL simulation and flight test.

If you have any question, please go to **https://rflysim.com** for your information.

Resource

All course PPTs, videos, and source code will be released on our website

https://rflysim.com/en

For more detailed content, please refer to the textbook:

Quan Quan, Xunhua Dai, Shuai Wang. *Multicopter Design and Control Practice*. Springer, 2020

https://www.springer.com/us/book/9789811531378

If you encounter any problems, please post question at Github page

https://github.com/RflySim/RflyExpCode/issues

If you are interested in RflySim advanced platform and courses for rapid development and testing of UAV Swarm/Vision/AI algorithms, please visit:

https://rflysim.com/en/4_Pro/Advanced.html

Thanks