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

Lesson 10 Set-point 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

(1) System characteristics in time domain

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

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

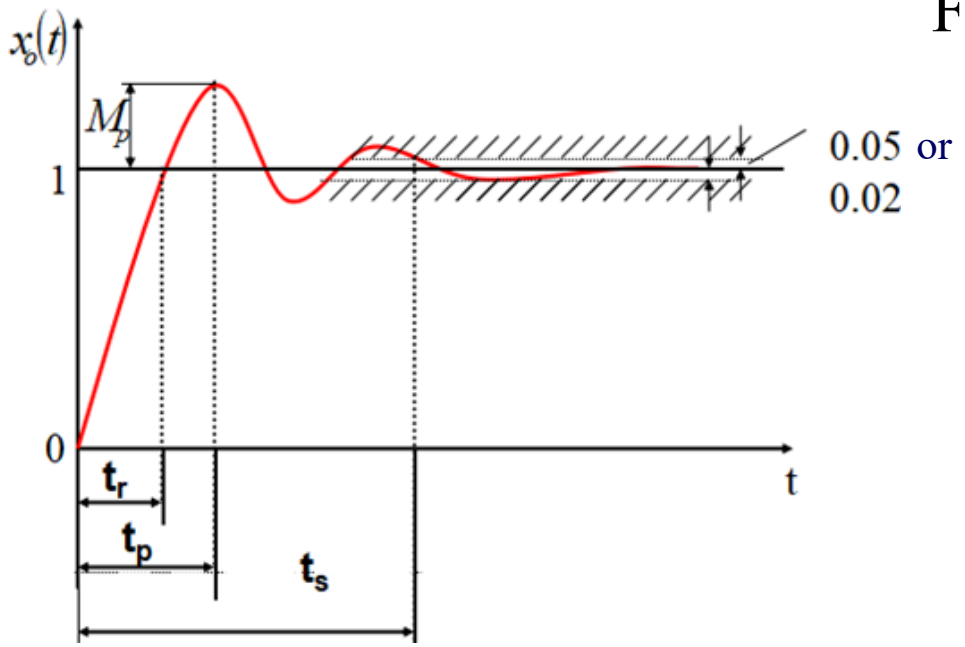


Figure: the step response of the second-order system

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

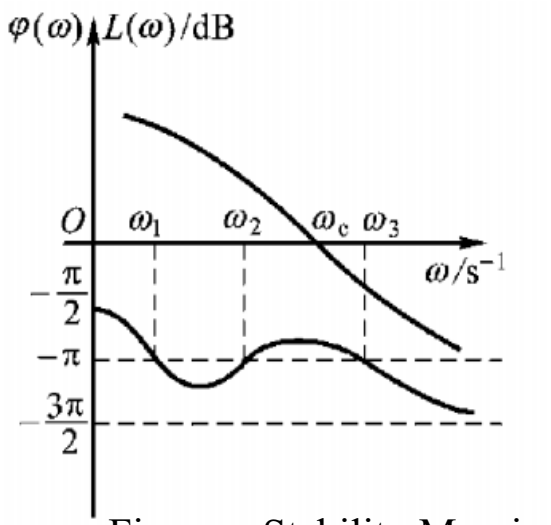
$$t_s=\frac{3.5}{\xi\omega_n} \quad \text{or} \quad t_s=\frac{4.5}{\xi\omega_n}$$

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

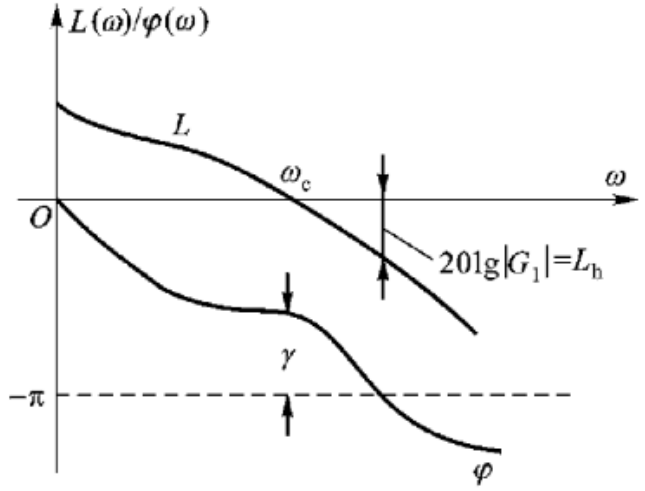


Figure: logarithm phase frequency curve

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

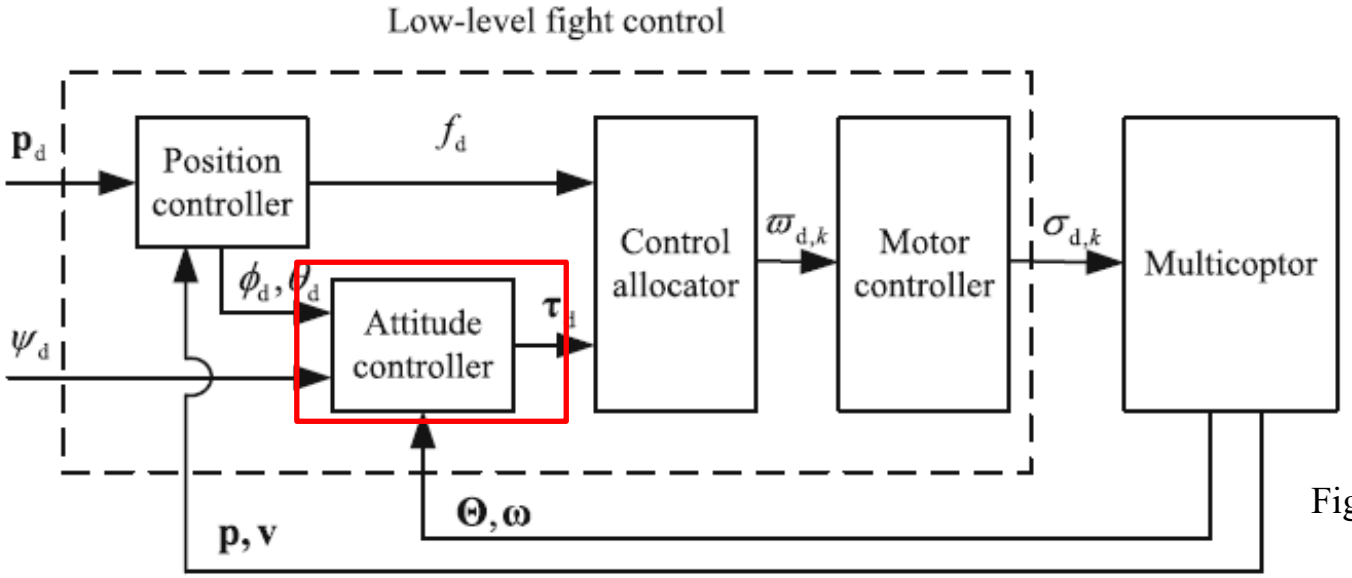


Figure: Closed-loop structure of a low-level fight control system for multicopters

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

Position Control

（1）Traditional PID Controller

- Horizontal position channel

$$\dot{\mathbf{p}}_{\mathrm{h}} = \mathbf{v}_{\mathrm{h}}$$

$$\dot{\mathbf{v}}_{\mathrm{h}} = -g\mathbf{A}_{\psi}\boldsymbol{\Theta}_{\mathrm{h}}$$

- Desired horizontal dynamics

$$\ddot{\mathbf{p}}_{\mathrm{h}} = \ddot{\mathbf{p}}_{\mathrm{hd}} - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{d}}\left(\dot{\mathbf{p}}_{\mathrm{h}} - \dot{\mathbf{p}}_{\mathrm{hd}}\right) - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{p}}\left(\mathbf{p}_{\mathrm{h}} - \mathbf{p}_{\mathrm{hd}}\right)$$

$$-\mathrm{g}\mathbf{A}_{\psi}\boldsymbol{\Theta}_{\mathrm{hd}} = \ddot{\mathbf{p}}_{\mathrm{hd}} - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{d}}\left(\dot{\mathbf{p}}_{\mathrm{h}} - \dot{\mathbf{p}}_{\mathrm{hd}}\right) - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{p}}\left(\mathbf{p}_{\mathrm{h}} - \mathbf{p}_{\mathrm{hd}}\right)$$

When considering the set-point control

$$\dot{\mathbf{p}}_{\mathrm{hd}} = \ddot{\mathbf{p}}_{\mathrm{hd}} = \mathbf{0}_{2\times 1}$$

$$\boldsymbol{\Theta}_{\mathrm{hd}} = -\mathrm{g}^{-1}\mathbf{A}_{\psi}^{-1}\left(\ddot{\mathbf{p}}_{\mathrm{hd}} - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{d}}\left(\dot{\mathbf{p}}_{\mathrm{h}} - \dot{\mathbf{p}}_{\mathrm{hd}}\right) - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{p}}\left(\mathbf{p}_{\mathrm{h}} - \mathbf{p}_{\mathrm{hd}}\right)\right)$$

- Where $\mathbf{K}_{(\cdot)}$ represents parameters.

Preliminary

Position Control

（1） Traditional PID Controller

- Altitude channel

$$\dot{p}_z = v_z$$

$$\dot{v}_z = g - \frac{f}{m}$$

- Desired altitude dynamics

$$\ddot{p}_z = \ddot{p}_{z_\mathrm{d}} - k_{p_z\mathrm{d}}\left(\dot{p}_z - \dot{p}_{z_\mathrm{d}}\right) - k_{p_z\mathrm{p}}\left(p_z - p_{z_\mathrm{d}}\right)$$

$$f_\mathrm{d} = mg - m\left(\ddot{p}_{z_\mathrm{d}} - k_{p_z\mathrm{d}}\left(\dot{p}_z - \dot{p}_{z_\mathrm{d}}\right) - k_{p_z\mathrm{p}}\left(p_z - p_{z_\mathrm{d}}\right)\right)$$

When considering the set-point control, $\dot{p}_{z_\mathrm{d}} = \ddot{p}_{z_\mathrm{d}} = 0$,so it becomes

$$f_\mathrm{d} = mg - m\left(-k_{p_z\mathrm{d}}\dot{p}_z - k_{p_z\mathrm{p}}\left(p_z - p_{z_\mathrm{d}}\right)\right)$$

Preliminary

☐ Position Control

（2）PID Controllers in Open Source Autopilots

1）Horizontal position control

■ In order to satisfy $\lim_{t\to\infty}\left\|\mathbf{e}_{\mathbf{p}_\mathrm{h}}(t)\right\|=0$,according to

Desired velocity

$$\dot{\mathbf{p}}_\mathrm{h}=\mathbf{v}_\mathrm{h} \Rightarrow \mathbf{v}_\mathrm{hd}=\mathbf{K}_{\mathbf{p}_\mathrm{h}}\left(\mathbf{p}_\mathrm{hd}-\mathbf{p}_\mathrm{h}\right)$$

Under the assumption that $\dot{\mathbf{p}}_\mathrm{hd}=0$, if

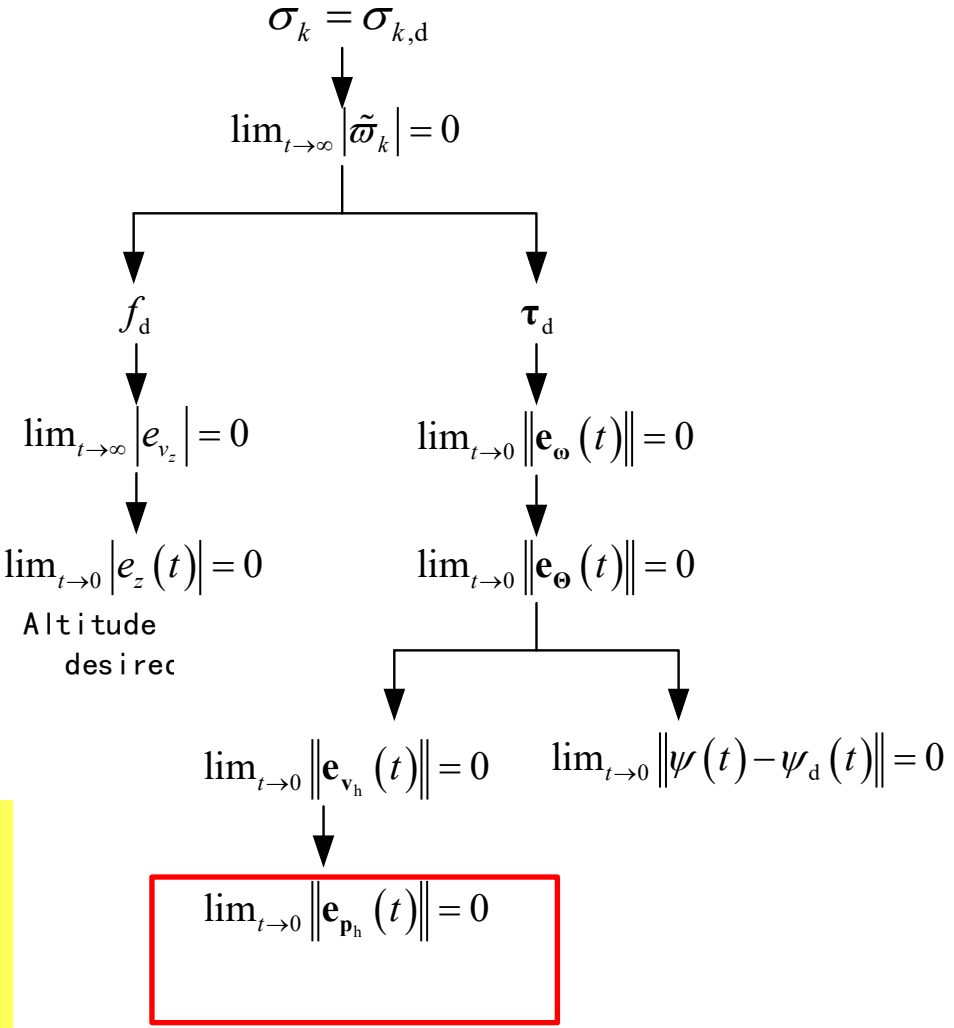
$$\lim_{t\to\infty}\left\|\mathbf{e}_{\mathbf{v}_\mathrm{h}}(t)\right\|=0$$

Then,

$$\lim_{t\to\infty}\left\|\mathbf{e}_{\mathbf{p}_\mathrm{h}}(t)\right\|=0$$

where $\mathbf{e}_{\mathbf{v}_\mathrm{h}}\triangleq\mathbf{v}_\mathrm{h}-\mathbf{v}_\mathrm{hd}$ 。

The velocity will reach the desired value as long as the position.



Preliminary

Position Control

(2) PID Controllers in Open Source Autopilots

1) Horizontal position channel

In order to satisfy $\lim_{t\to\infty}\left\|\mathbf{e}_{\mathbf{v}_\mathrm{h}}(t)\right\|=0$, according to

$$\dot{\mathbf{v}}_\mathrm{h}=-g\mathbf{A}_\psi\boldsymbol{\Theta}_\mathrm{h}$$

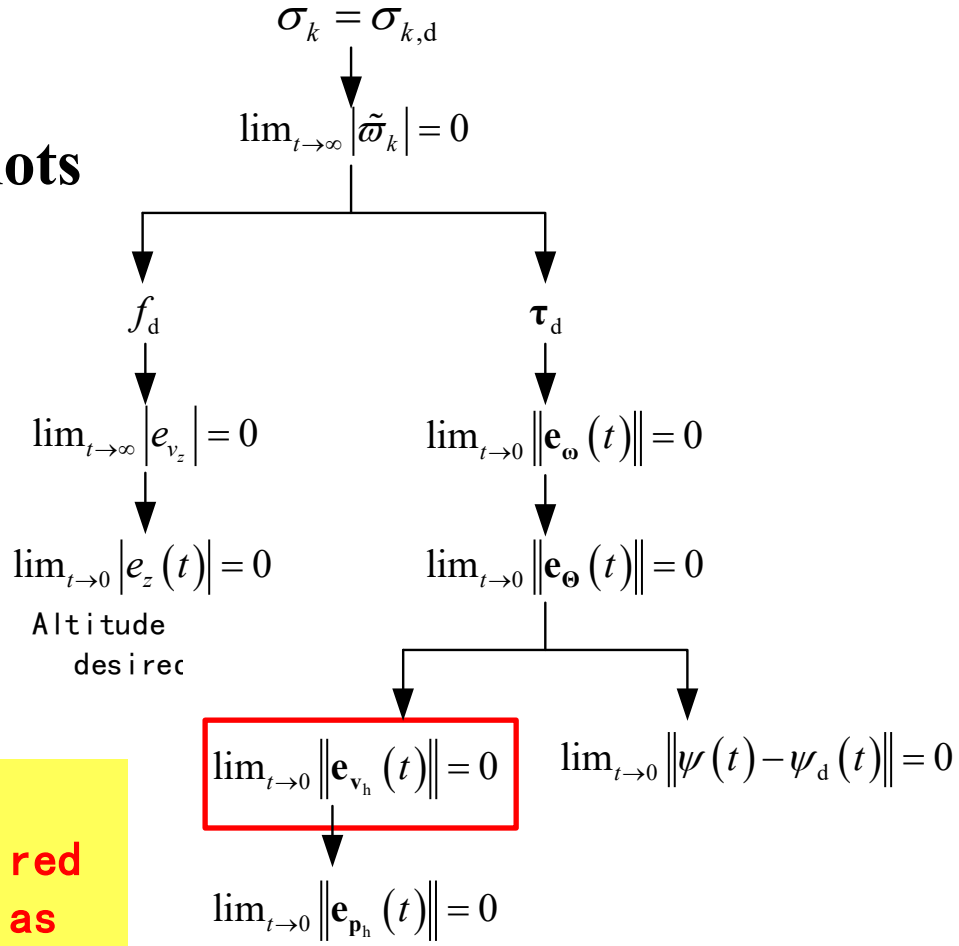
$$-g\mathbf{A}_\psi\boldsymbol{\Theta}_\mathrm{hd}=-\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{p}}\mathbf{e}_{\mathbf{v}_\mathrm{h}}-\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{i}}\int\mathbf{e}_{\mathbf{v}_\mathrm{h}}-\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{d}}\dot{\mathbf{e}}_{\mathbf{v}_\mathrm{h}}$$

$$\boldsymbol{\Theta}_\mathrm{hd}=g^{-1}\mathbf{A}_\psi^{-1}\left(\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{p}}\mathbf{e}_{\mathbf{v}_\mathrm{h}}+\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{i}}\int\mathbf{e}_{\mathbf{v}_\mathrm{h}}+\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{d}}\dot{\mathbf{e}}_{\mathbf{v}_\mathrm{h}}\right)$$

If, $\lim_{t\to\infty}\left\|\boldsymbol{\Theta}_\mathrm{h}(t)-\boldsymbol{\Theta}_\mathrm{hd}(t)\right\|=0$

Then, $\lim_{t\to\infty}\left\|\mathbf{e}_{\mathbf{v}_\mathrm{h}}(t)\right\|=0$

The angle will reach the desired value as long as the velocity.



Preliminary

Position Control

（2） PID Controllers in Open Source Autopilots

2）Altitude channel

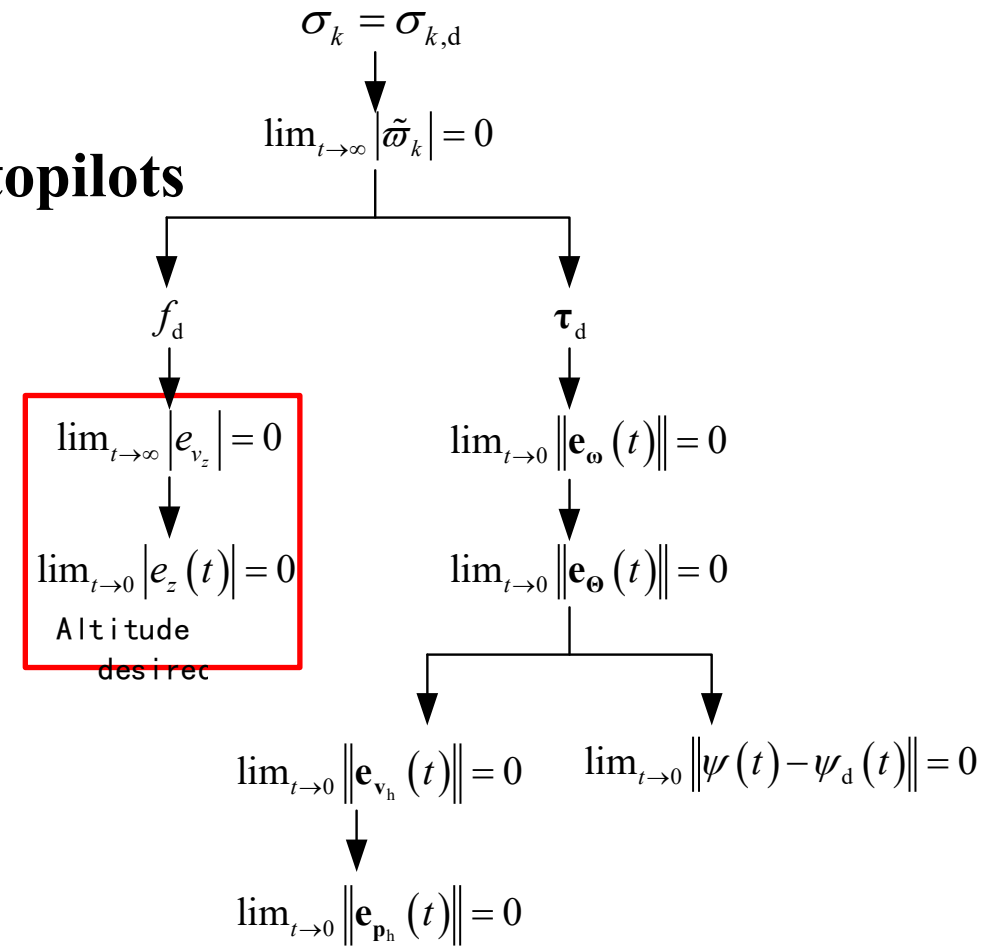
Similar to the horizontal position channel,

The altitude channel controller is

$$v_{z\mathrm{d}} = -k_{p_z}\left(p_z - p_{z\mathrm{d}}\right)$$

$$f_{\mathrm{d}} = m\left(g + k_{v_z\mathrm{p}}e_{v_z} + k_{v_z\mathrm{i}}\int e_{v_z} + k_{v_z\mathrm{d}}\dot{e}_{v_z}\right)$$

The vertical velocity will reach the desired value as long as the altitude.



Experiment Preliminary

☐ Position Control

（3）PID Controller with Saturation

Traditional PID controller

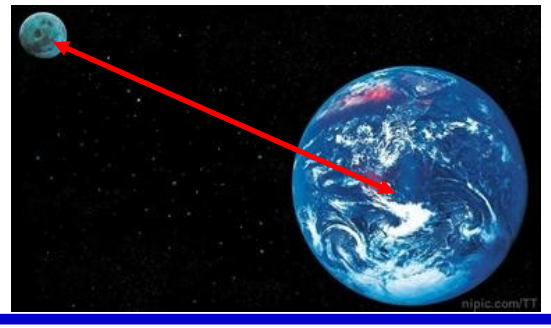
$$\boldsymbol{\Theta}_{\mathrm{hd}} = -g^{-1}\mathbf{A}_{\psi}^{-1}\left(\ddot{\mathbf{p}}_{\mathrm{hd}} - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{d}}\left(\dot{\mathbf{p}}_{\mathrm{h}} - \dot{\mathbf{p}}_{\mathrm{hd}}\right) - \mathbf{K}_{\mathbf{p}_{\mathrm{h}}\mathrm{p}}\left(\mathbf{p}_{\mathrm{h}} - \mathbf{p}_{\mathrm{hd}}\right)\right)$$

Then design of the Controller make no Sense.

PID Controllers in Open Source Autopilots

$$\boldsymbol{\Theta}_{\mathrm{hd}} = g^{-1}\mathbf{A}_{\psi}^{-1}\left(\mathbf{K}_{\mathbf{v}_{\mathrm{h}}\mathrm{p}}\mathbf{e}_{\mathbf{v}_{\mathrm{h}}} + \mathbf{K}_{\mathbf{v}_{\mathrm{h}}\mathrm{i}}\int \mathbf{e}_{\mathbf{v}_{\mathrm{h}}} + \mathbf{K}_{\mathbf{v}_{\mathrm{h}}\mathrm{d}}\dot{\mathbf{e}}_{\mathbf{v}_{\mathrm{h}}}\right)$$

The position error is large



The position error is large

$$\boldsymbol{\Theta}_{\mathrm{hd}} \gg 2\pi$$

The small-angle assumption is violated.

It is necessary to add a controller with saturation

Experiment Preliminary

Position Control

（3） PID Controller with Saturation

The PID controller used in open source autopilot is rewritten as

$$\mathbf{e}_{\mathbf{v}_\mathrm{h}} = \mathrm{sat}_{\mathrm{gd}}\left(\mathbf{v}_\mathrm{h} - \mathbf{v}_{\mathrm{hd}}, a_1\right)$$

$$\boldsymbol{\Theta}_{\mathrm{hd}} = \mathrm{sat}_{\mathrm{gd}}\left(\mathrm{g}^{-1}\mathbf{A}_{\psi}^{-1}\left(\mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{p}}\mathbf{e}_{\mathbf{v}_\mathrm{h}} + \mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{i}}\int \mathbf{e}_{\mathbf{v}_\mathrm{h}} + \mathbf{K}_{\mathbf{v}_\mathrm{h}\mathrm{d}}\dot{\mathbf{e}}_{\mathbf{v}_\mathrm{h}}\right), a_2\right)$$

where $a_1, a_2 \geq 0$ .The direction-guaranteed saturation function $\mathrm{sat}_{\mathrm{gd}}(\mathbf{u}, a)$ is defined as

$$\mathrm{sat}_{\mathrm{gd}}(\mathbf{u},\, a) \triangleq \begin{cases} \mathbf{u}, & \|\mathbf{u}\|_\infty \leq a \\ a\,\dfrac{\mathbf{u}}{\|\mathbf{u}\|_\infty}, & \|\mathbf{u}\|_\infty > a \end{cases}$$

The difference between the direction-guaranteed saturation $\mathrm{sat}_{\mathrm{gd}}(\mathbf{u}, a)$ and the traditional saturation function $\mathrm{sat}(\mathbf{u}, a)$: the direction-guaranteed saturation function can not only confine that the absolute value of each element of the final vector is not greater than a, but also guarantee the direction to be the same as that of x.

Experiment Preliminary

Position Control

(3) PID Controller with Saturation

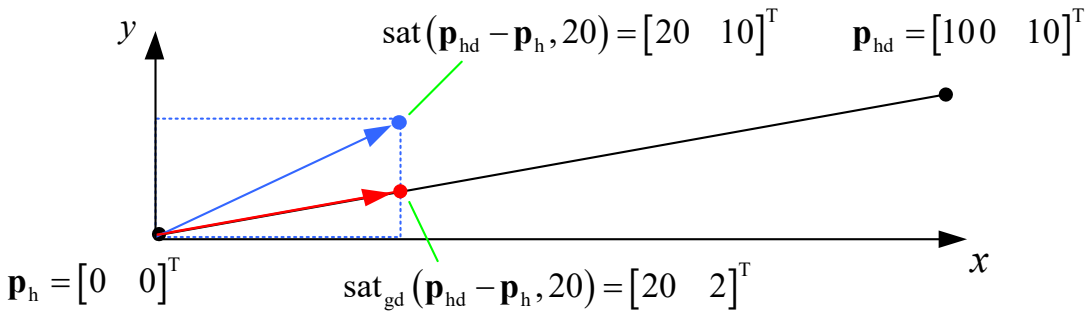


Figure: Comparison of the results of two Saturation functions

Traditional saturation function

$$\mathrm{sat}(\mathbf{x},a)\triangleq\begin{bmatrix}\mathrm{sat}(x_1,a)\\ \vdots\\ \mathrm{sat}(x_n,a)\end{bmatrix},\ \mathrm{sat}(x_k,a)\triangleq\begin{cases}x_k & ,|x_k|\le a\\ a\cdot\mathrm{sign}(x_k), & |x_k|>a\end{cases}$$

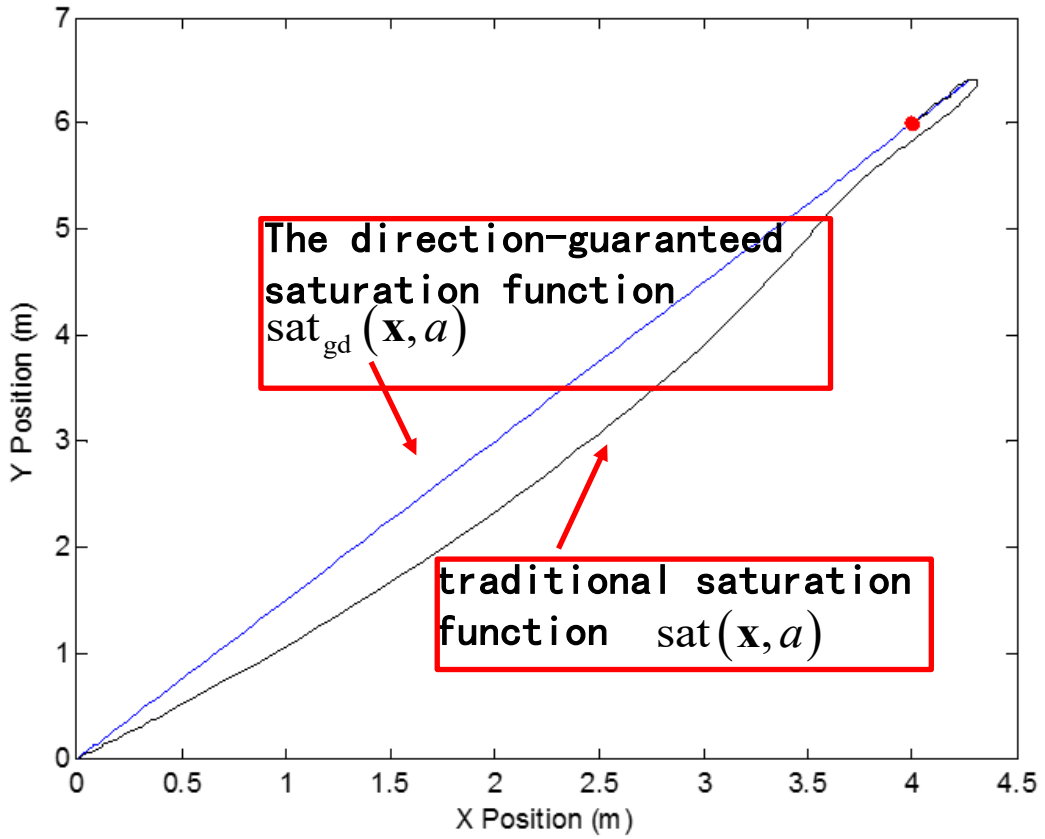


Figure: Comparison of the results of two Saturation functions

The direction-guaranteed saturation function can guarantee that the multicopter flies along a straight line, but the traditional saturation function cannot.

Experiment Preliminary

□ Position Control

（3） PID Controller with Saturation

In order to avoid a throttle command out of range, the saturation needs to be considered as well. Thus, the traditional PID controller becomes

$$f_{\mathrm{d}} = \mathrm{sat}_{\mathrm{gd}}\left(m\left(g + k_{p_z\mathrm{d}}\dot{p}_z + k_{p_z\mathrm{p}}\left(p_z - p_{z_{\mathrm{d}}}\right)\right), a_3\right)$$

Where $a_3 \in \mathbb{R}_+$. Similarly, the PID controller design used in the open source autopilots (11.29) is rewritten as

$$e_{v_z} = \mathrm{sat}_{\mathrm{gd}}\left(v_z - v_{z\mathrm{d}}, a_4\right)$$

$$f_{\mathrm{d}} = \mathrm{sat}_{\mathrm{gd}}\left(m\left(g + k_{v_z\mathrm{p}}e_{v_z} + k_{v_z\mathrm{i}}\int e_{v_z} + k_{v_z\mathrm{d}}\dot{e}_{v_z}\right), a_5\right)$$

Where $a_4, a_5 \in \mathbb{R}_+$. For a scale, the direction-guaranteed saturation function is the same as the traditional saturation function. 。

Experiment 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.

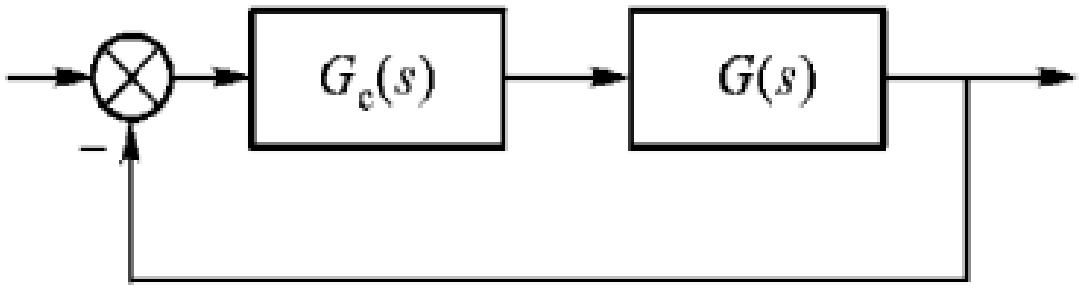


Figure: System with series compensation

Experiment 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:

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

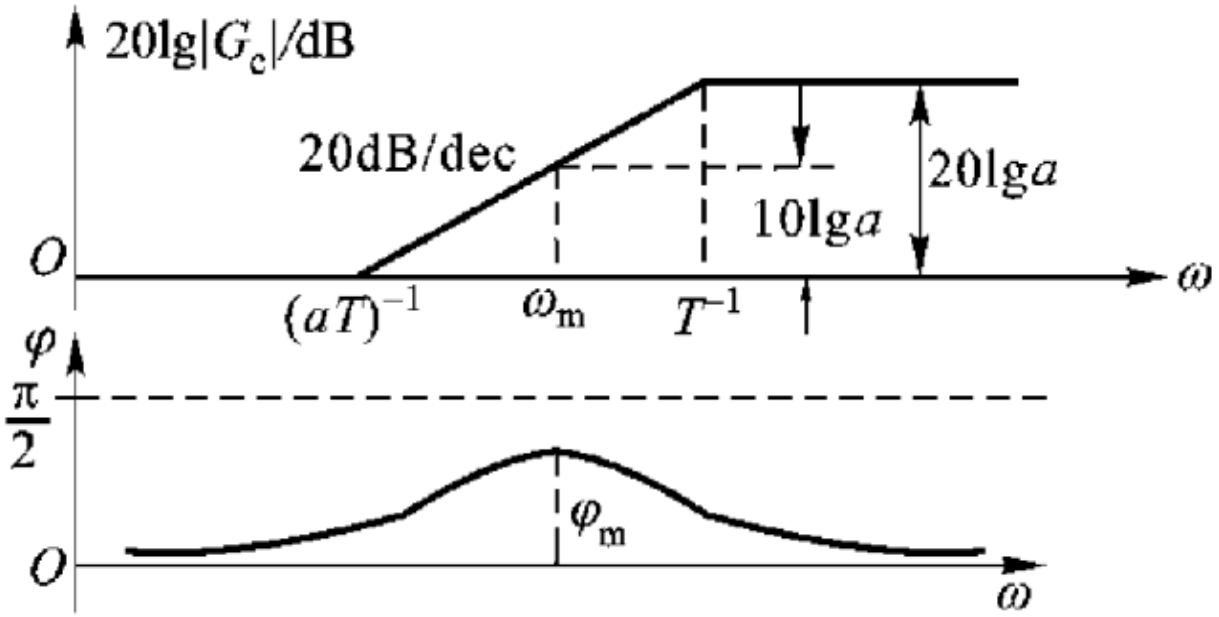


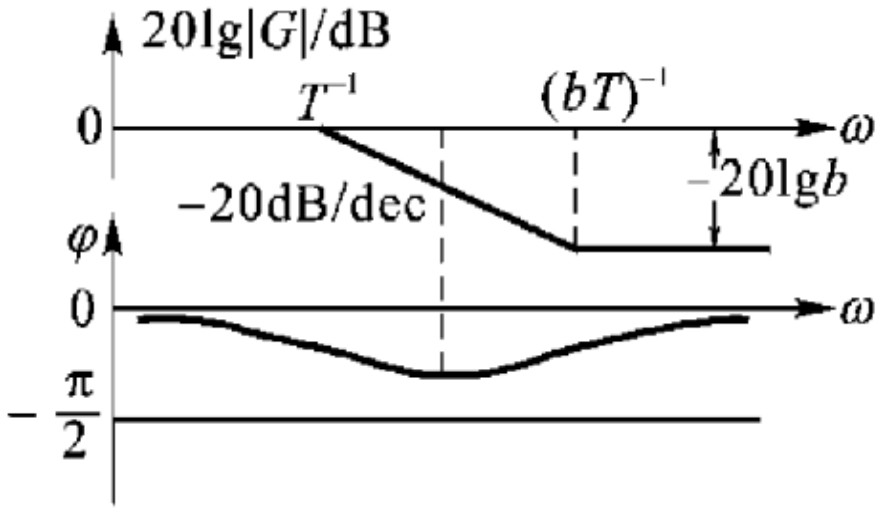
Figure: Phase-Lead compensation curve

Experiment Preliminary

☐ The control system compensation

（2）Phase-lag compensation

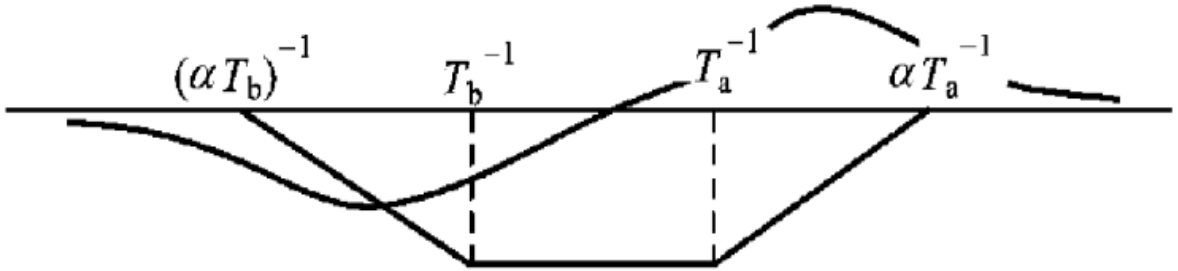
$$G_c(s)=\frac{1+bTs}{1+Ts}(b<1)$$



（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 $bT_1>aT_2$



Experiment Preliminary

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

Experiment Objectives

Things to prepare

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

（2）Software: MATLAB 2017b and above, Simulink-based Controller Design and Simulation Platform, HIL(Hardware in the loop) Simulation Platform, Experiment Instruction Package “e6.1”（https://flyeval.com/course）.

Objectives

(1)Repeat the Simulink simulation of a quadcopter to analyze the channel decoupling between the control along $o_b x_b$ and $o_b y_b$ axes ;

（2）Sweep the open-loop position control system to obtain the Bode plot and further analyze the stability margin of the closed-loop position control system;

（3）Perform the HIL simulation.

Basic Experiment

☐ Simulation procedure

（1） Step1:SIL simulation - channel decouping

1） Parameter Initialization

Run the file “e5/e5.1/Init_control.m” to initialize the parameters. Next, the Simulink file“AttitudeControl_Sim” will open automatically as shown on the right.

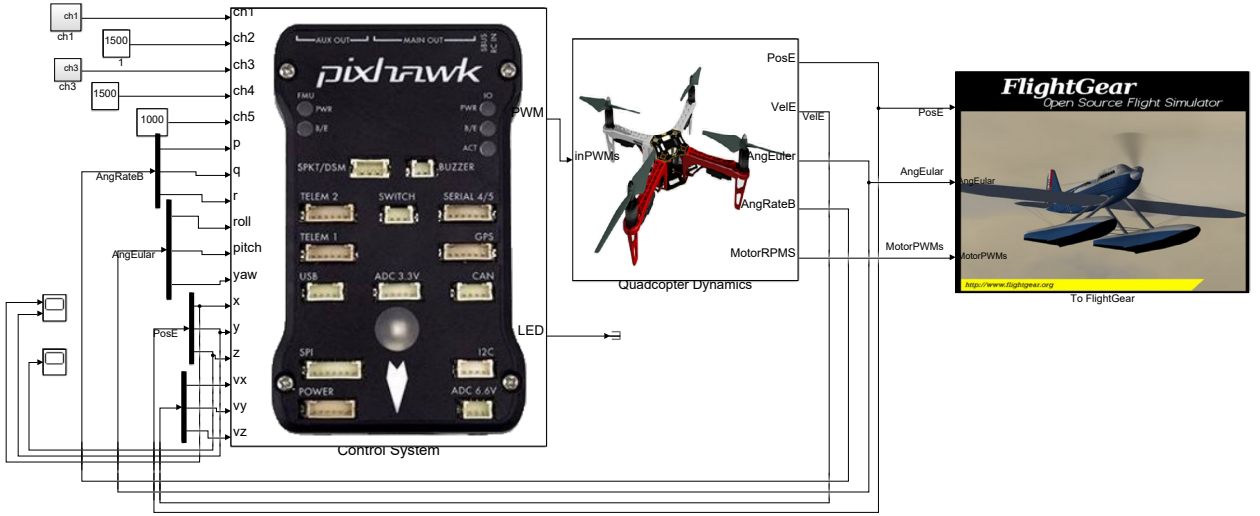


Figure: Simulink model “PoaitionControl_Sim.slx”

Basic Experiment

☐ Simulation procedure

2）Run the simulation

Open the file “FlightGear-F450” and click on the Simulink “Run” button to run. 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 $o_e y_e$ axis

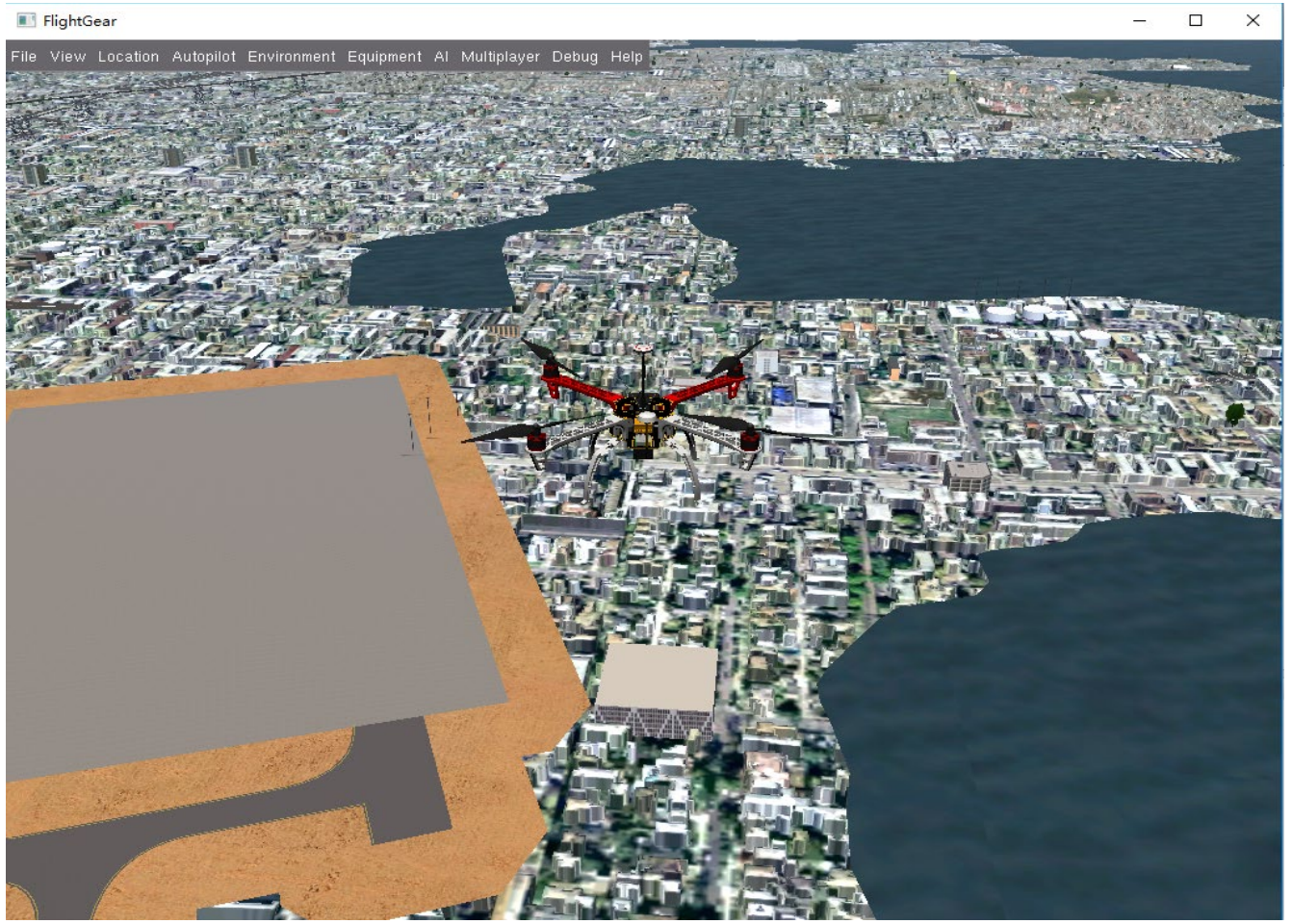
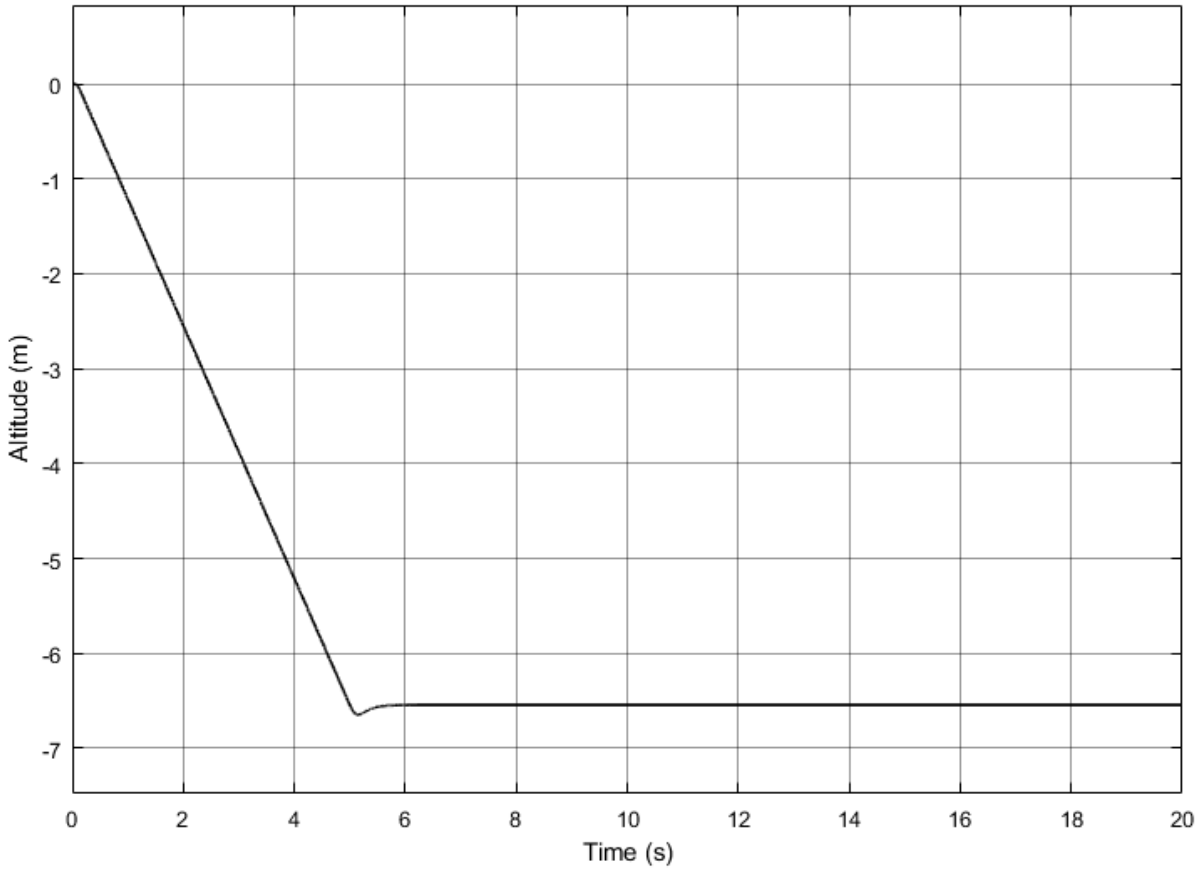


Figure: Quadcopter in FlightGear

Simulation procedure

3） Simulation results

The change of the altitude is shown on the right.



Figure； Altitude response

Simulation procedure

4）Channel decoupling analysis

The resultant horizontal position of the quadcopter is shown on the right. It can be observed that position control along the $o_b y_b$ axis does not change the position along the $o_b x_b$ axis. This implies that control actions along the $o_b x_b$ and $o_b y_b$ axes have been decoupled.

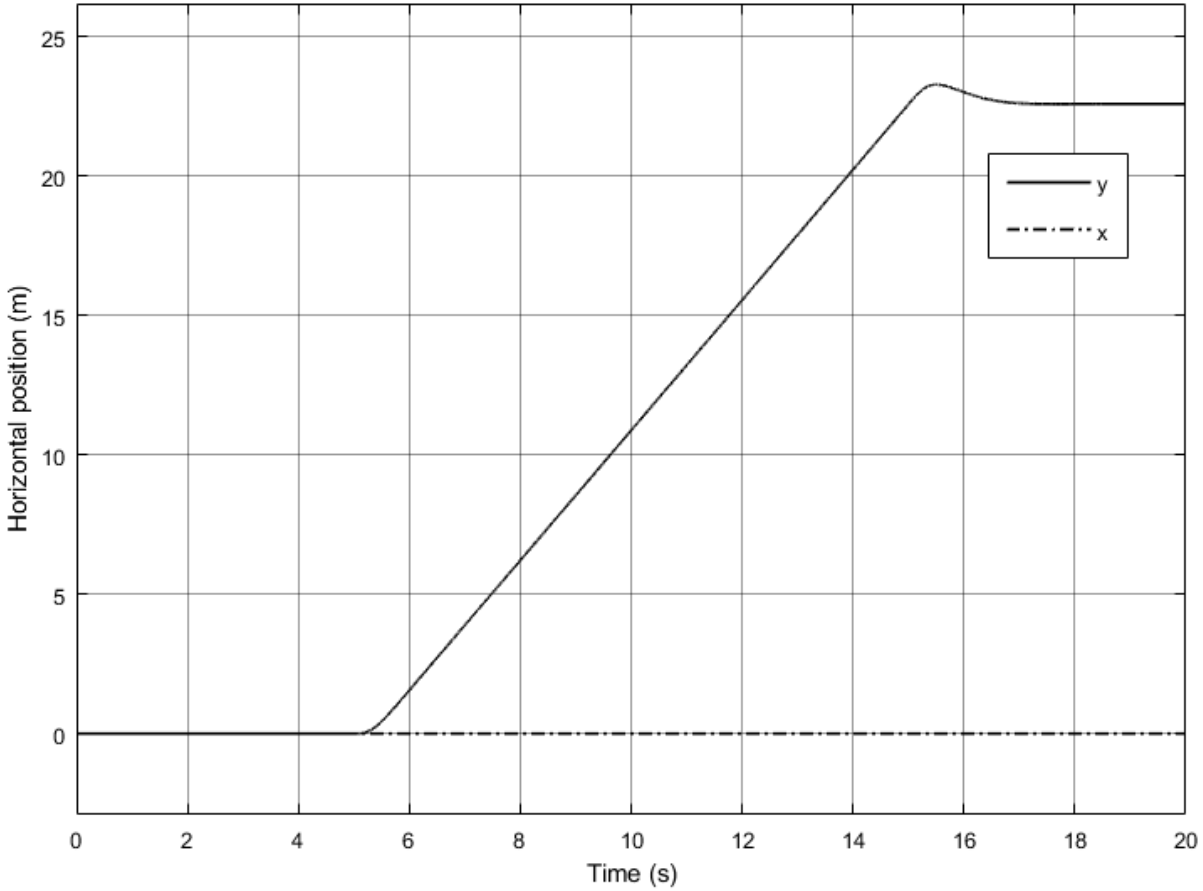


Figure: Horizontal position response

Basic Experiment

Simulation procedure

（2）Step2:SIL simulation - stability

1）Run the file “e6/e6.1/tune/Init_control.m”to initialize the parameters, and then the Simulink file “PosControl_tune.slx” is opened automatically.

Open the model “Control System”-“position_control” of the Simulink file and specify the input and output signals for the Bode plot.

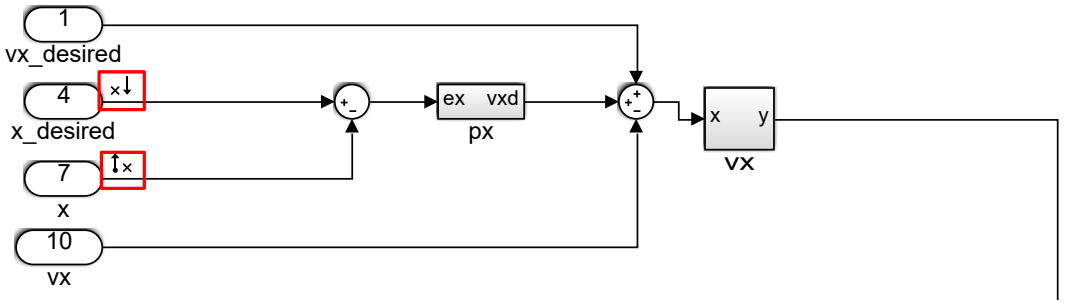


Figure: Specify signals as input and output

Basic Experiment

Simulation procedure

2）Select “Analysis”- “Control Design”- “Linear Analysis” in the top menu.

3）From the context menu,select “Linear Analysis” and click “Bode” to get the Bode plot.

4）Right click the curve and select “Characteristics” - “All Stability Margins”. It is observed that the gain margin is 15.3dB at a frequency of 3.97rad/s;the phase margin is 65.5， at a frequency of 1.04rad/s.

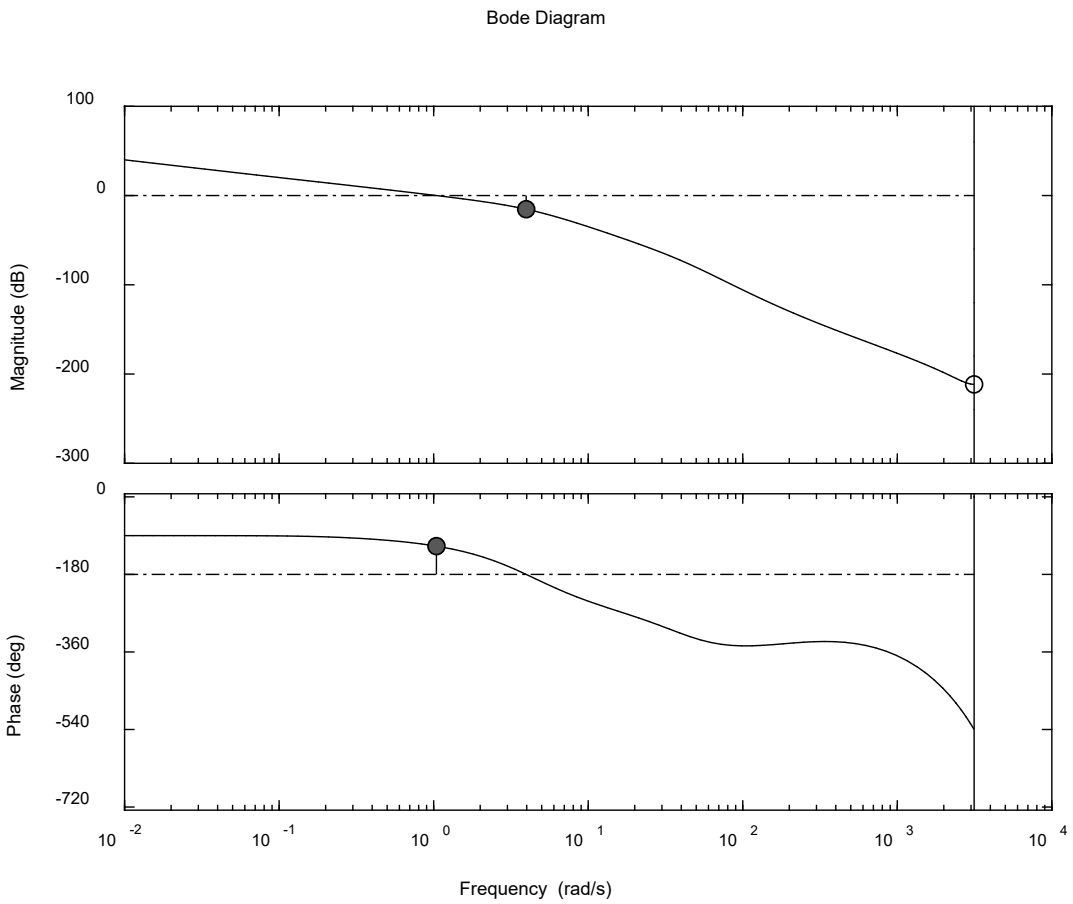


Figure: Open-loop Bode plot of x-axis channel control system

Basic Experiment

Simulation procedure

（3）Step3: HIL simulation

1）Open Simulink file for HIL

Run the file “e6/e6.1/HIL/Init_control.m” to initialize the parameters， and then the Simulink file “PosControl_HIL.slx” is opened automatically as shown on the right.

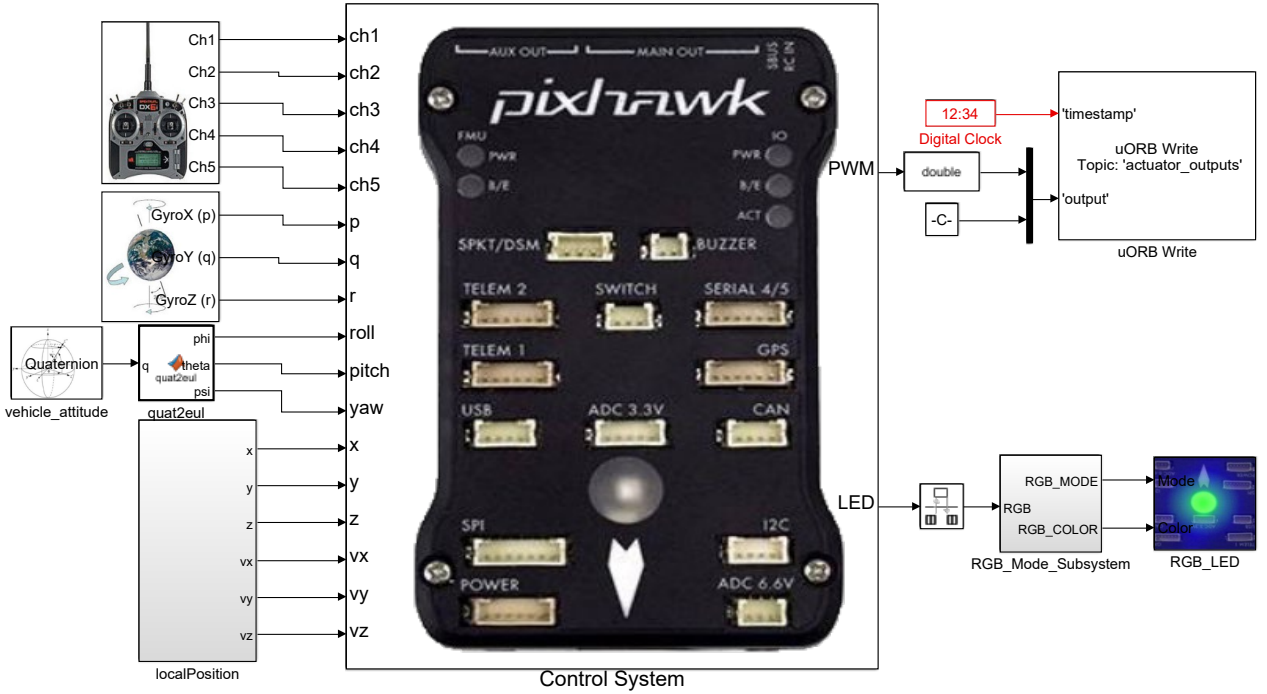


Figure: Simulink model “AttitudeControlHIL.slx”

Basic Experiment

☐ Simulation 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

Simulation 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.

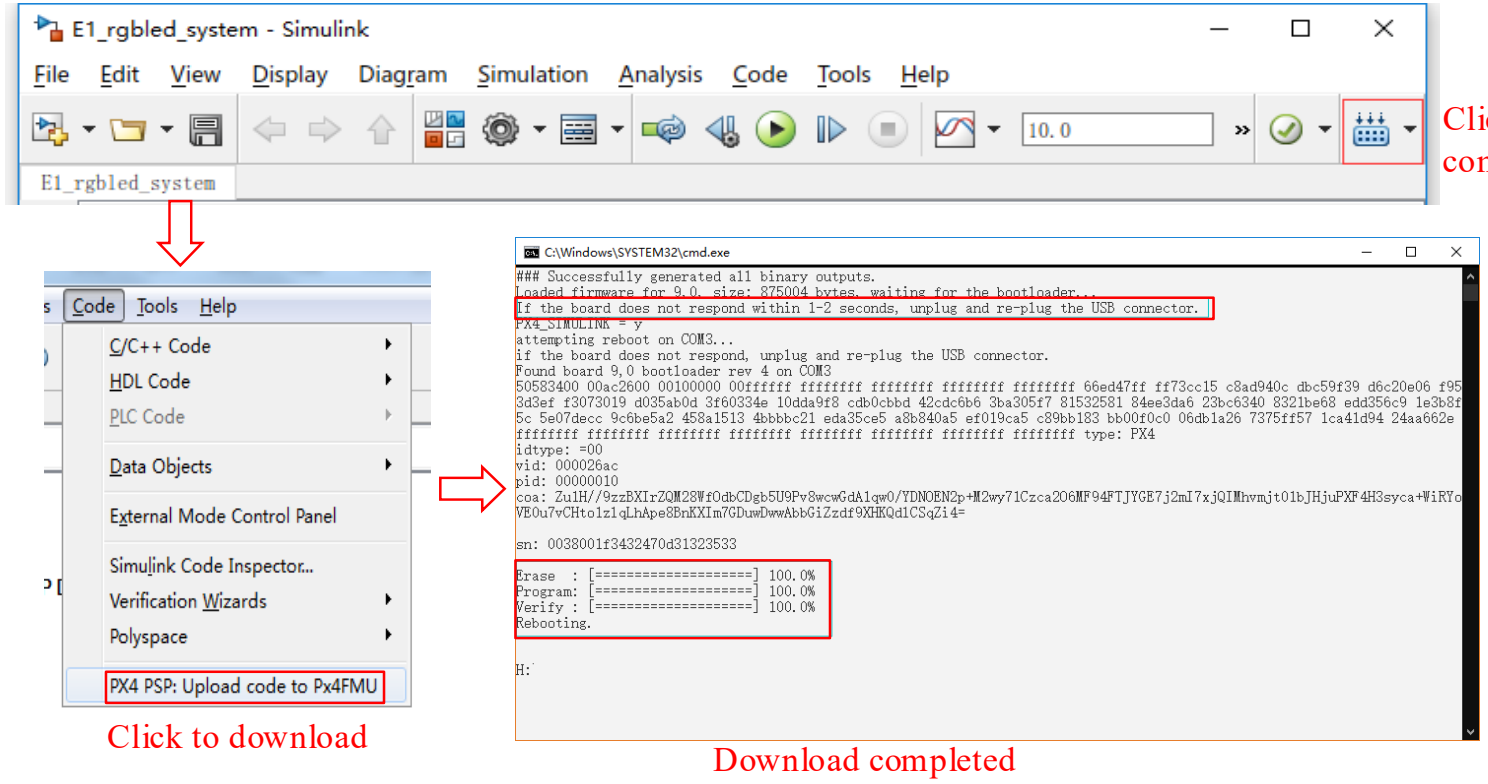


Figure: Code compilation and upload process

Basic Experiment

☐ Simulation 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.

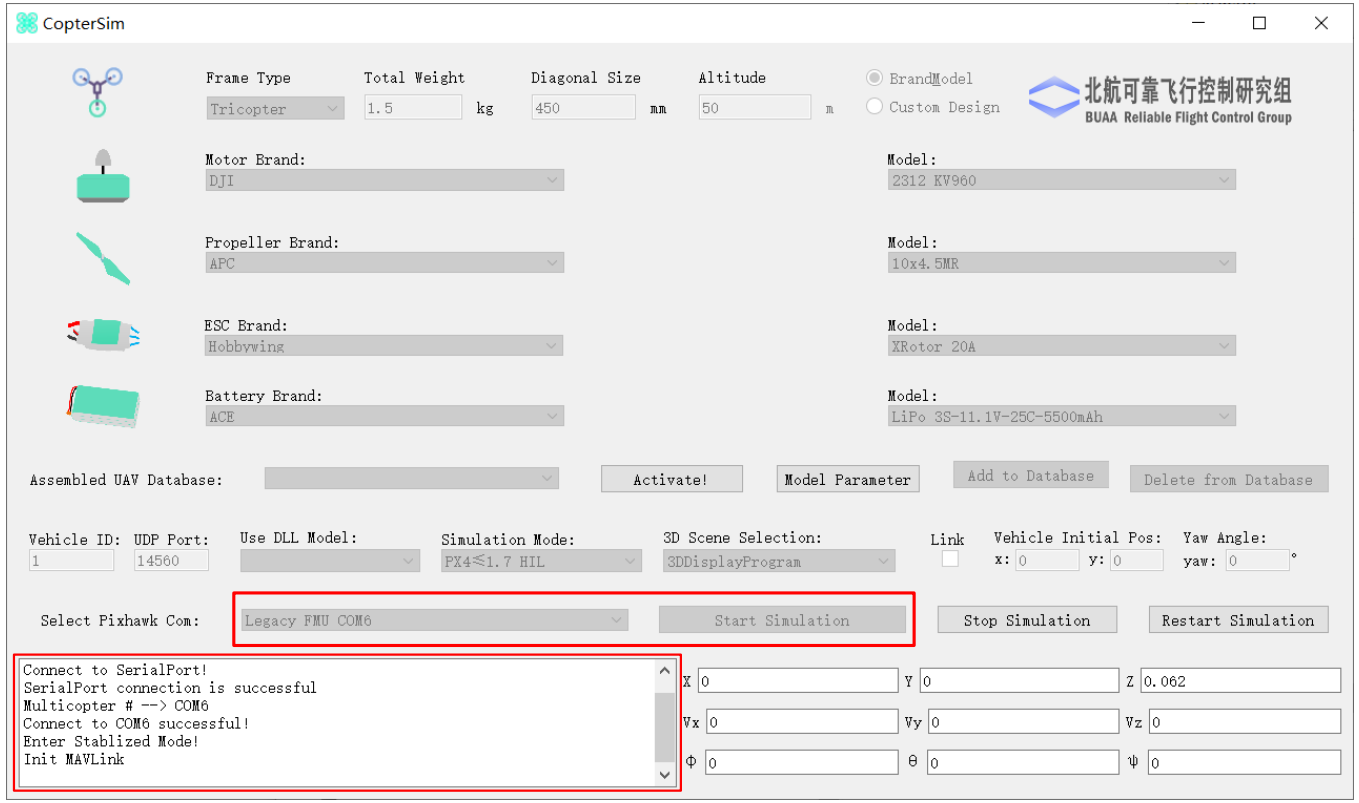


Figure: User interface of CopterSim

Basic Experiment

Simulation 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 transmitter7. The quadcopter can hover and fly at a specified speed. When all control sticks are in the middle position, the quadcopter will keep hovering.

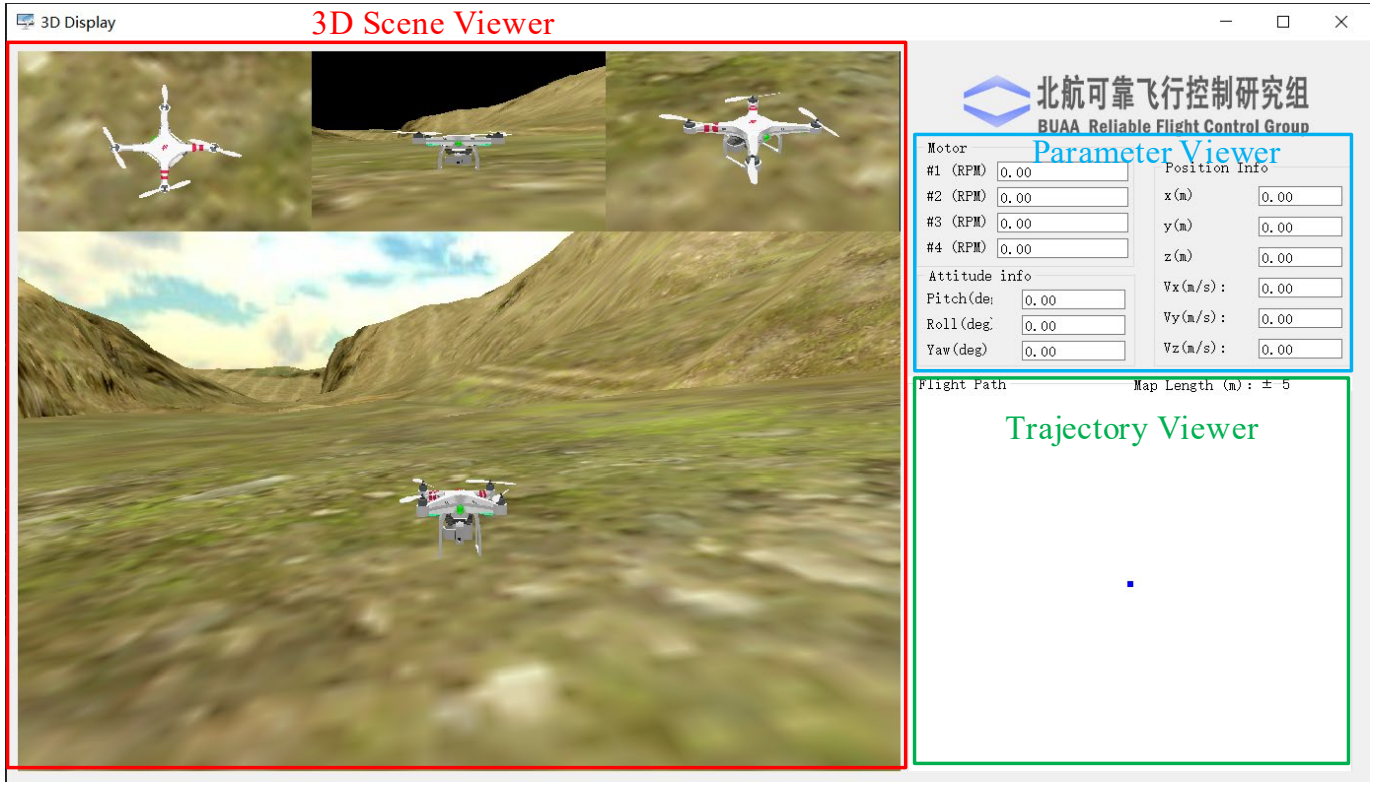


Figure: User interface of 3DDisplay

Experiment Objectives

Things to prepare

(1)Software: MATLAB 2017b and above, Simulink-based Controller Design and Simulation Platform, Experimental Instruction Package “e6.2”（https://flyeval.com/course）.

Objectives

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

（2） Using the obtained satisfied parameters, sweep the system to draw the Bode plot and analyze its stability margin.

Analysis Experiment

Simulation Procedure

（1） Step1:Initial model setup

The steps to adjust PID parameters are similar to those used for attitude control. First, adjust the altitude channel and then adjust the horizontal position channel. Furthermore, for each channel, first adjust the velocity control loop and then adjust the position loop. The necessary file can be found in the “e6/e6.2/tune” folder. Because the PID adjusting methods for the channel control loops are similar, the PID parameters for velocity along the *oexe* axis are adjusted here as an example. First, let the quadcopter hover at the initial altitude of 100m by setting the throttle value to 0.6085 and the initial speed of four motors to 557.1420rad/s. Modify the corresponding parameters in the file “Init_control.m” as shown on the right.

```
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

Simulation Procedure

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

Open the model “Control System” - “position_control” in the file “e6/e6.2/tune/PosControl_tune.slx”. Replace “vx_desired” with a step input. Later, configure a step response and “vx” to “Enable Data logging” to get the step response of the velocity control loop, as shown on the right.

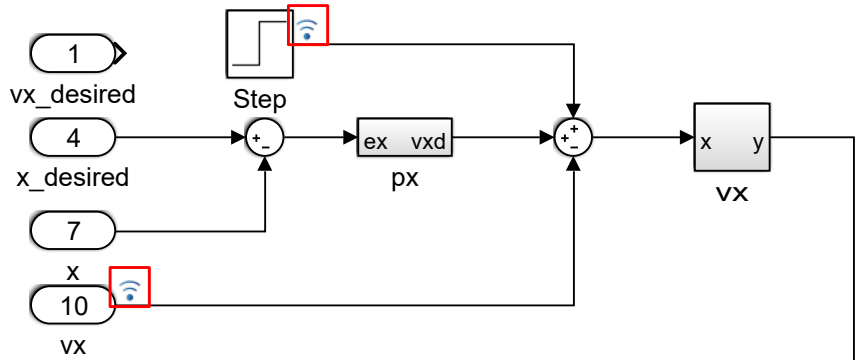


Figure: Velocity control loop model

Analysis Experiment

Simulation Procedure

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

Modify the PID parameters corresponding to the velocity control loop in the file “Init_control.m” . First, adjust the proportional term parameter and set the integral and derivative term parameters to 0. After that, run the file “Init_control.m” (you must run the file each time to update any changes if you change them). Click on the Simulink“Run” button to view responses in the“Simulation Data Inspector”. The proportional term parameter gradually increases from a small value to a large value, i.e., the variable “Kvxp”in the file “Init_control.m” increases.

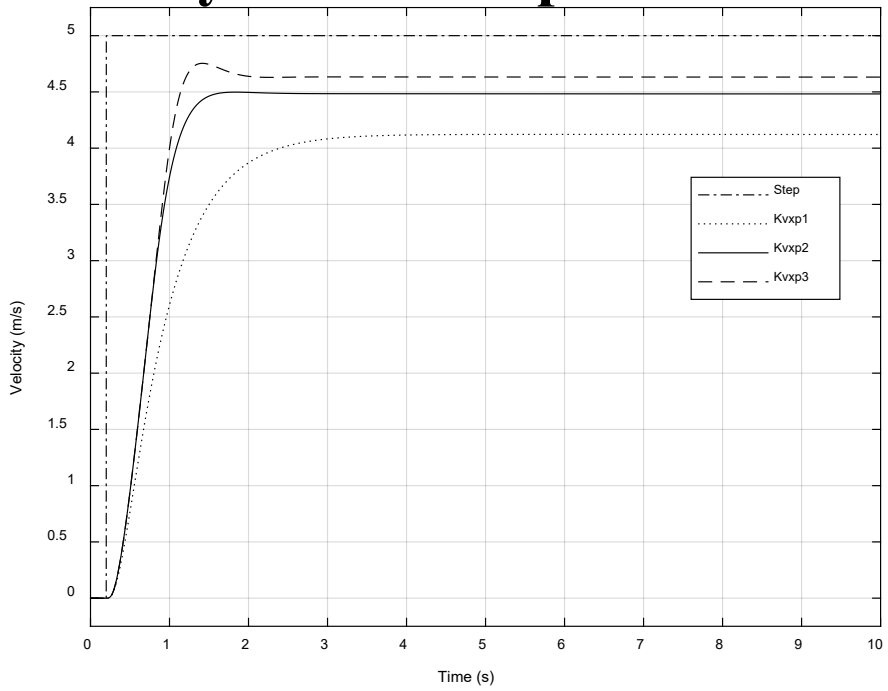


Figure: Step response of velocity along the $o_e x_e$ axis channel with different proportional term parameters

Analysis Experiment

☐ Simulation Procedure

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

Later, adjust the integral and derivative term parameters, i.e.,“Kvxi” and “Kvxd” in the file “Init_control.m”. Finally, fine tune the proportional term parameter and the resulting PID parameters are

Kvxp=2.5;

Kvxi=0.4;

Kvxd=0.01;

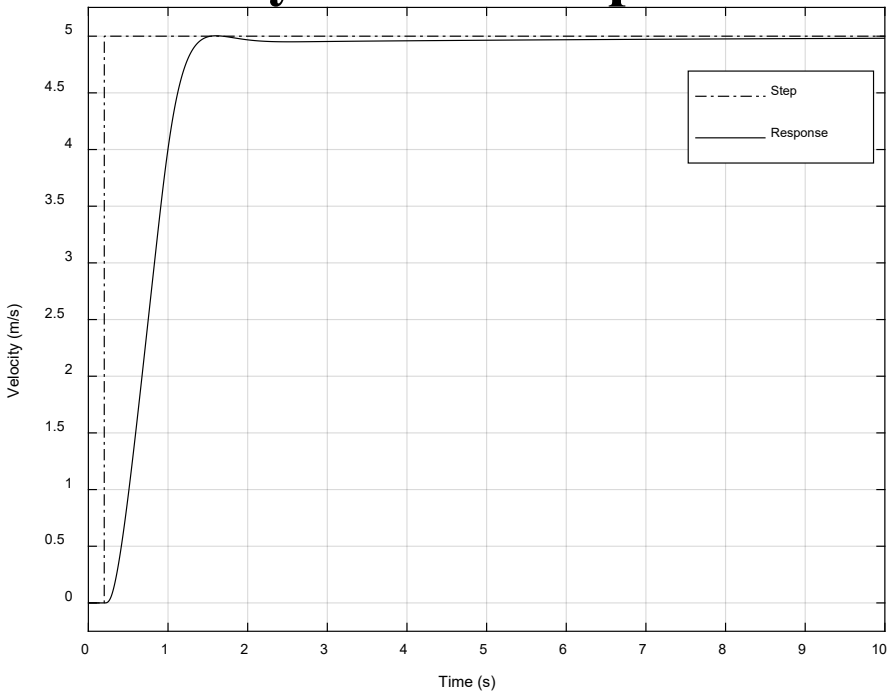


Figure: Step response of velocity along $o_e x_e$ axis channel with a group of satisfied PID parameters

Analysis Experiment

Simulation Procedure

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

Adjust the proportional term parameter for the position control loop. Using the obtained position control loop parameters in Step 2, replace “x_desired” with a step input, and set the step input and “x” in the “PosControl_tune.slx” file to “Enable Data Logging”.

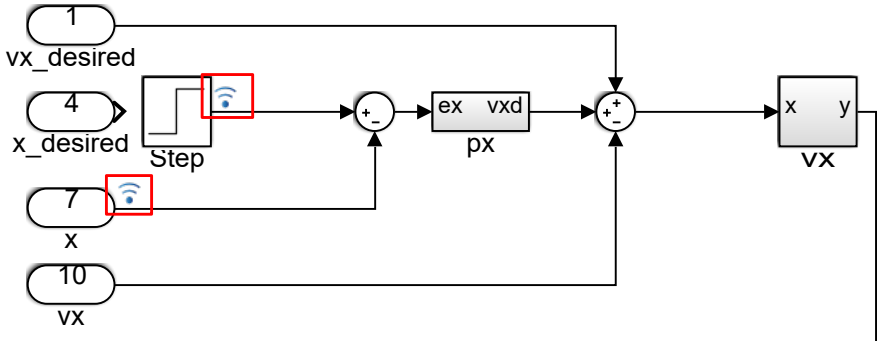


Figure: Position control loop along the $o_e x_e$ axis

Analysis Experiment

Simulation Procedure

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

Increase the proportional term parameter “Kpxp” in the file “Init_control.m” gradually, and observe the step response in“Simulation Data Inspector”. The final selected parameters are

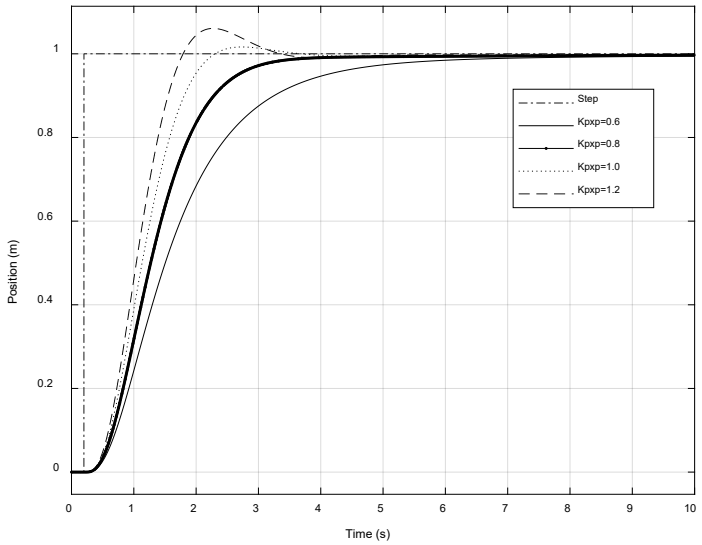


Figure: Step response of position control loop along $o_e x_e$ axis with different proportional term parameters

Kpxp=1.0;

Kvxp=2.5;

Kvxi=0.4;

Kvxd=0.01;

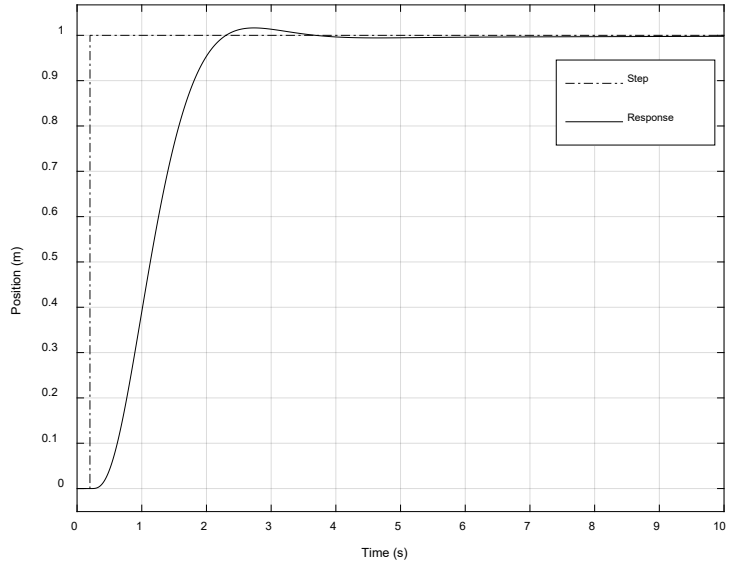


Figure: Step response of position control loop along $o_e x_e$ axis with different proportional term parameters

Analysis Experiment

☐ Simulation Procedure

（4）Step4: Sweep to get the Bode plot

Specify input and output signals for the Bode plot. Specify “x_desired” as “Open-loop Input”, and then specify “x” as “Open-loop Output”. Using these parameters, the Bode plot can be drawn, as shown on the right.

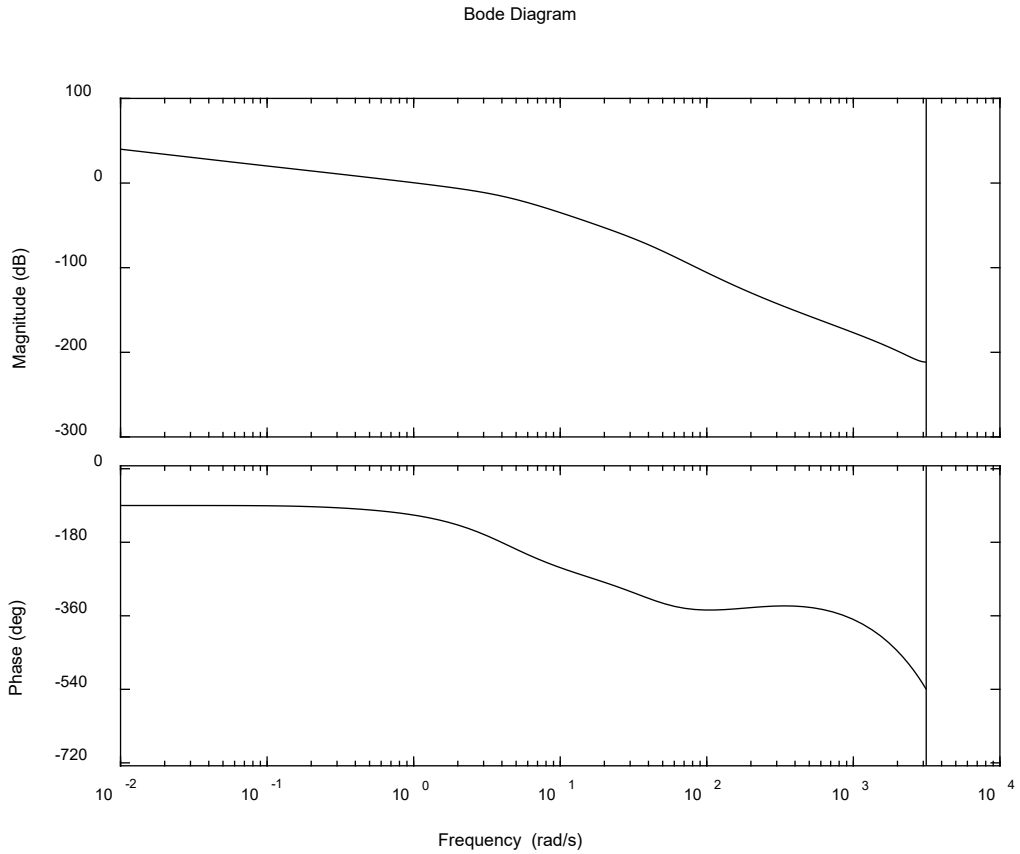


Figure: Bode plot of the open-loop position Control loop along the $o_e x_e$ axis

Analysis Experiment

☐ Remark

（1）The premise of the frequency domain response is to set the quadcopter in equilibrium so that

it hovers. Hence, it is important to set the quadcopter at equilibrium.

（2）Find out the input and output of the system, and then specify the input and output signals correctly. Select the output signal line as “Open-loop output” when testing the open-loop system, and “Output Measurement” when testing the closed-loop system. For details, please refer to the document https://ww2.mathworks.cn/help/slcontrol/ug/specify-portion-of-model-to-linearize-in-simulink-model.html.

Design Experiment

☐ Experimental Objectives

■ Things to prepare

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

2）Software: MATLAB 2017b and above, Simulink-based Controller Design and Simulation Platform, HIL Simulation Platform, Experiment Instruction Package “e6.3”（https://flyeval.com/course）.

■ Objectives

1）Establish the transfer function for the position control channel, and design a compensator for the existing controller using MATLAB “ControlSystemDesigner” in the velocity control loop to satisfy the conditions of step response steady-state error $e_{rss} \leq 0.01$， phase margin > 75°, and cut-off frequency > 2.0 rad/s. The position control loop satisfies that the cut-off frequency is >1 rad/s and phase margin is > 60°;

2) Perform SIL simulation and HIL simulation experiment with the designed controller;

3) Use the designed controller to perform outdoor flight test experiment.

Design Experiment

☐ Design Procedure

（1） Step1: Simplify the overall structure

Consider the x-axis channel for example. The simplified model is shown below.

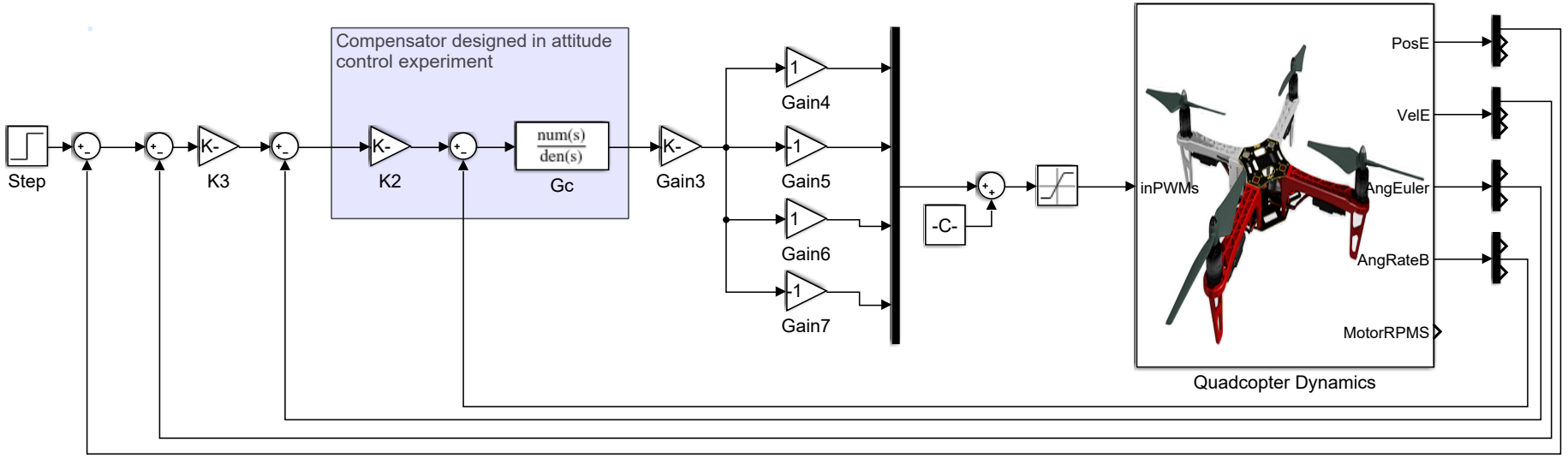


Figure: Simulink model ”PosControl_tune.slx”

Design Experiment

☐ Design Procedure

（2） Step2: Velocity control loop analysis

The input is the desired velocity and the output is the velocity. Specify signals as input and output.

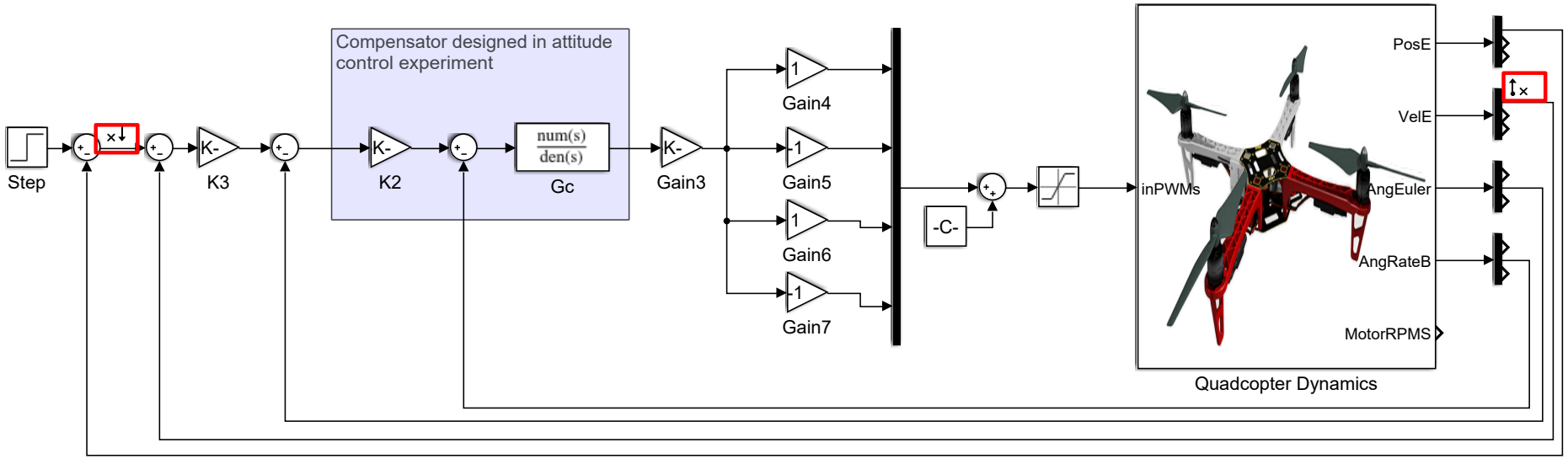


Figure: Specify signals as input and output

Design Experiment

Design Procedure

（3） Step3: Obtain the transfer function of the x-axis channel

After the Bode plot is obtained, a variable, namely “linsys1”, will appear in the “Linear Analysis Workspace”. The transfer function can be obtained by the operation shown on the right.

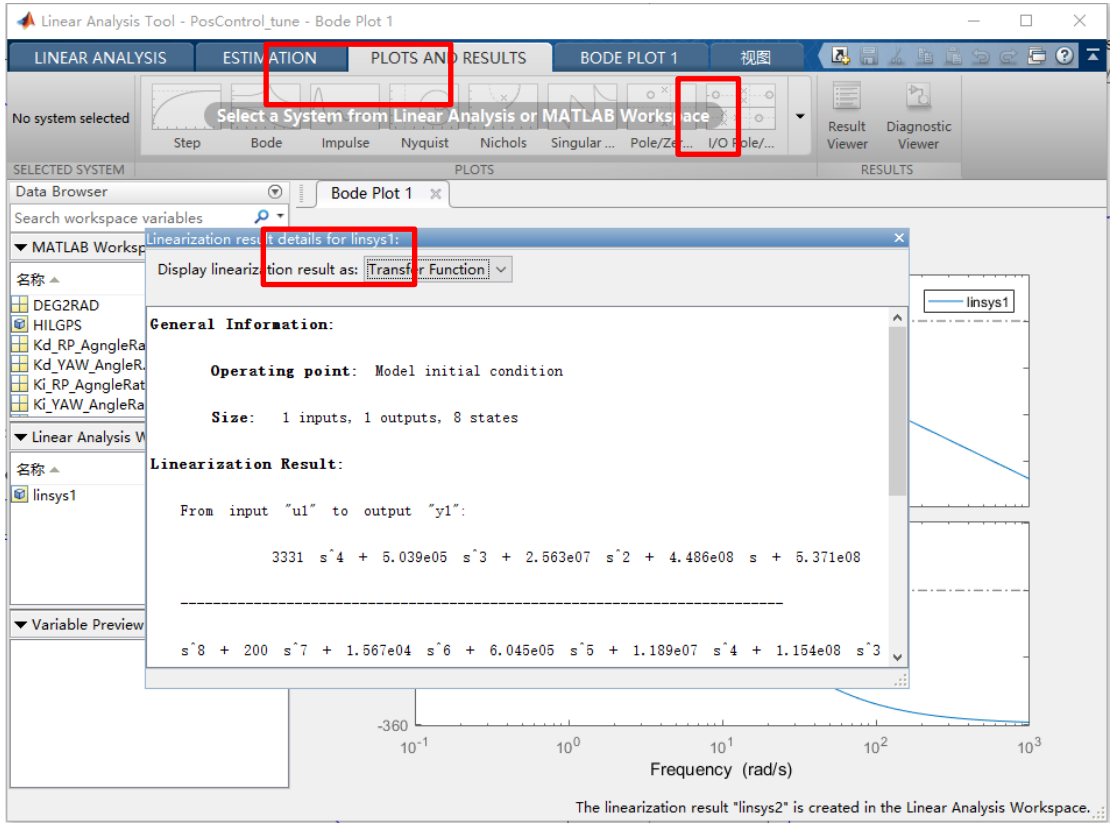


Figure: Transfer function

$$\frac{3330.9(s+1.29)}{(s+5.214e-07)(s+1.253)(s+33.92)(s^2+14.87s+101.1)}$$

Simplify

$$\frac{3331s^4+5.039e05s^3+2.563e07s^2+4.486e08s+5.371e08}{s^8+200s^7+1.567e04s^6+6.045e05s^5+1.189e07s^4+1.154e08s^3+5.557e08s^2+5.371e08s+280.1}$$

Design Experiment

(4) Step4: Use toolbox to design the compensator

Based on the transfer function, establish a ".m" file, then the "Control System Design" can be used to design the compensator.

```
1  num=[3331 5.039e05 2.563e07 4.486e08 5.371e08];
2  den=[1 200 1.567e04 6.045e05 1.189e07  ...
3  1.154e08 5.557e08 5.371e08 280.1];
4  G=tf(num,den);
5  controlSystemDesigner('bode',G);
```

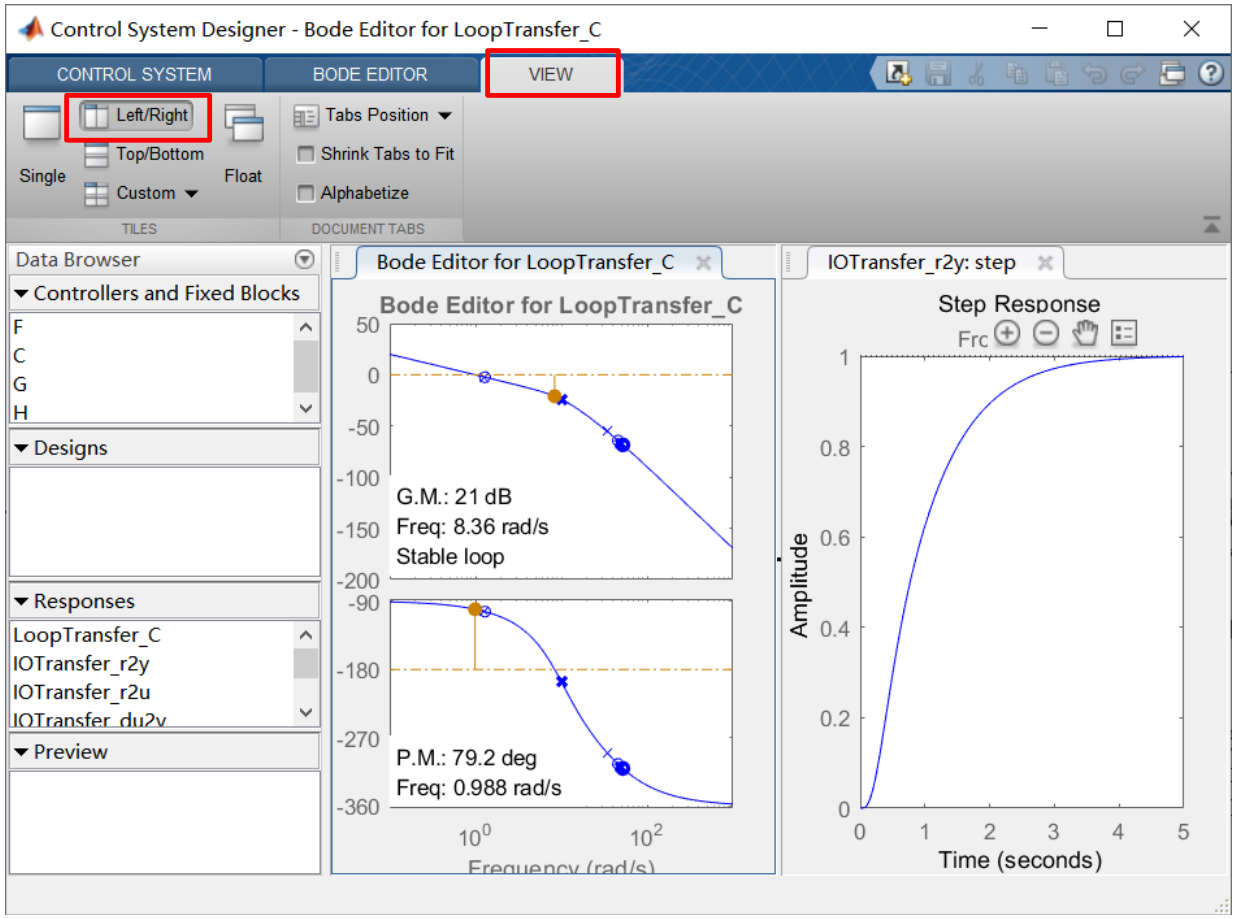
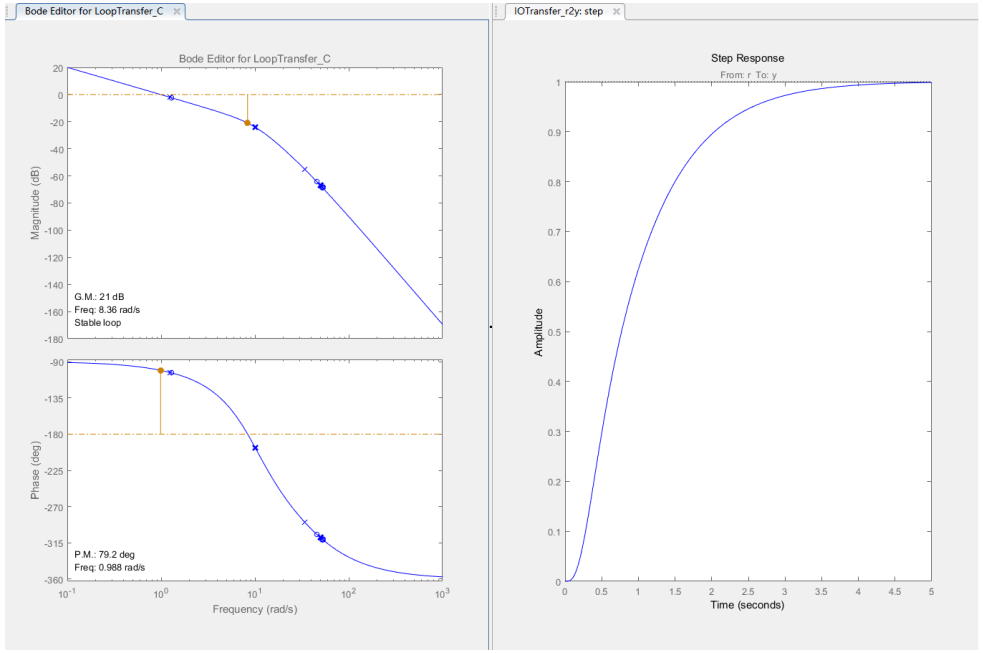


Figure: Control system design based on Bode plot

Design Experiment

（5） Step5: Use toolbox to design the compensator

It can be observed that the transient response is slow. Drag the curve in the Bode plot up to increase the open-loop gain so that the transient response becomes rapid.

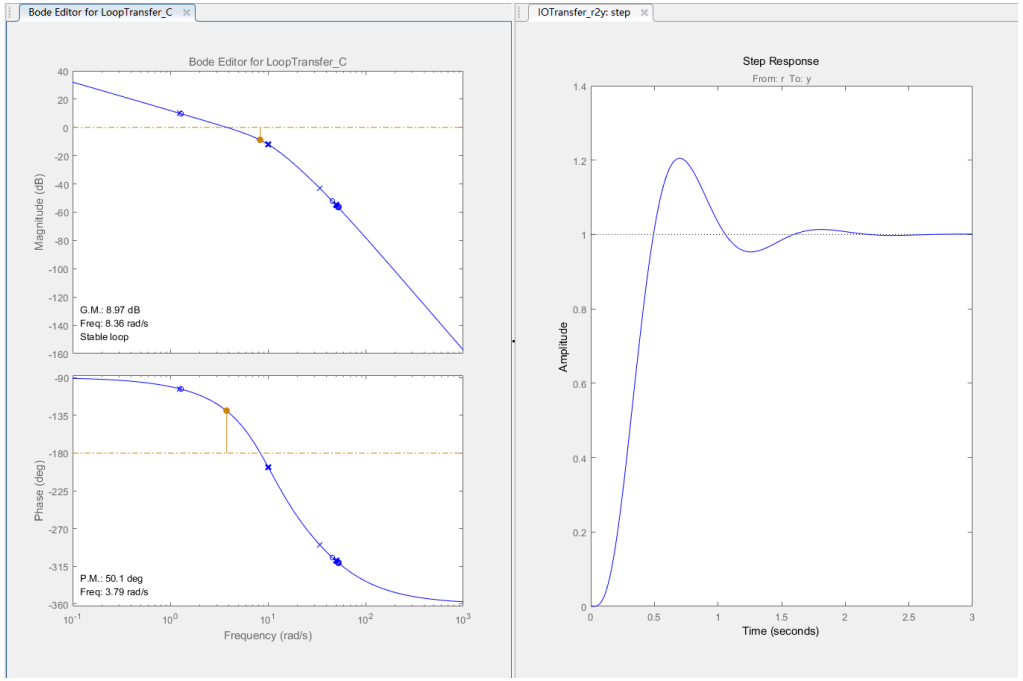


Design Experiment

（5） Step5: Use toolbox to design the compensator

Later, as shown on the right, the response time reduces but an overshoot appears, and the phase margin is 50.1. These

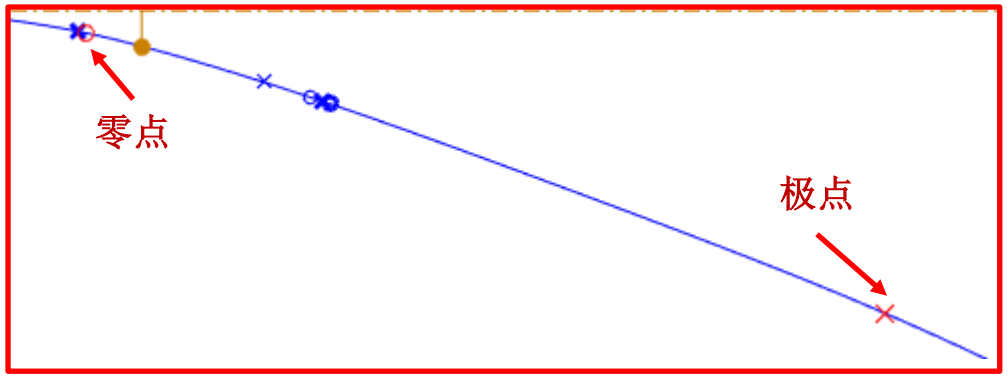
do not meet the requirement.



Design Experiment

(5) Step5: Use toolbox to design the compensator

Consider adding a lead compensator to increase the phase margin, and further increase the cut-off frequency and response speed. The procedure is as follows. In the Bode plot, right click and select “add Pole/Zero” - “Lead”.



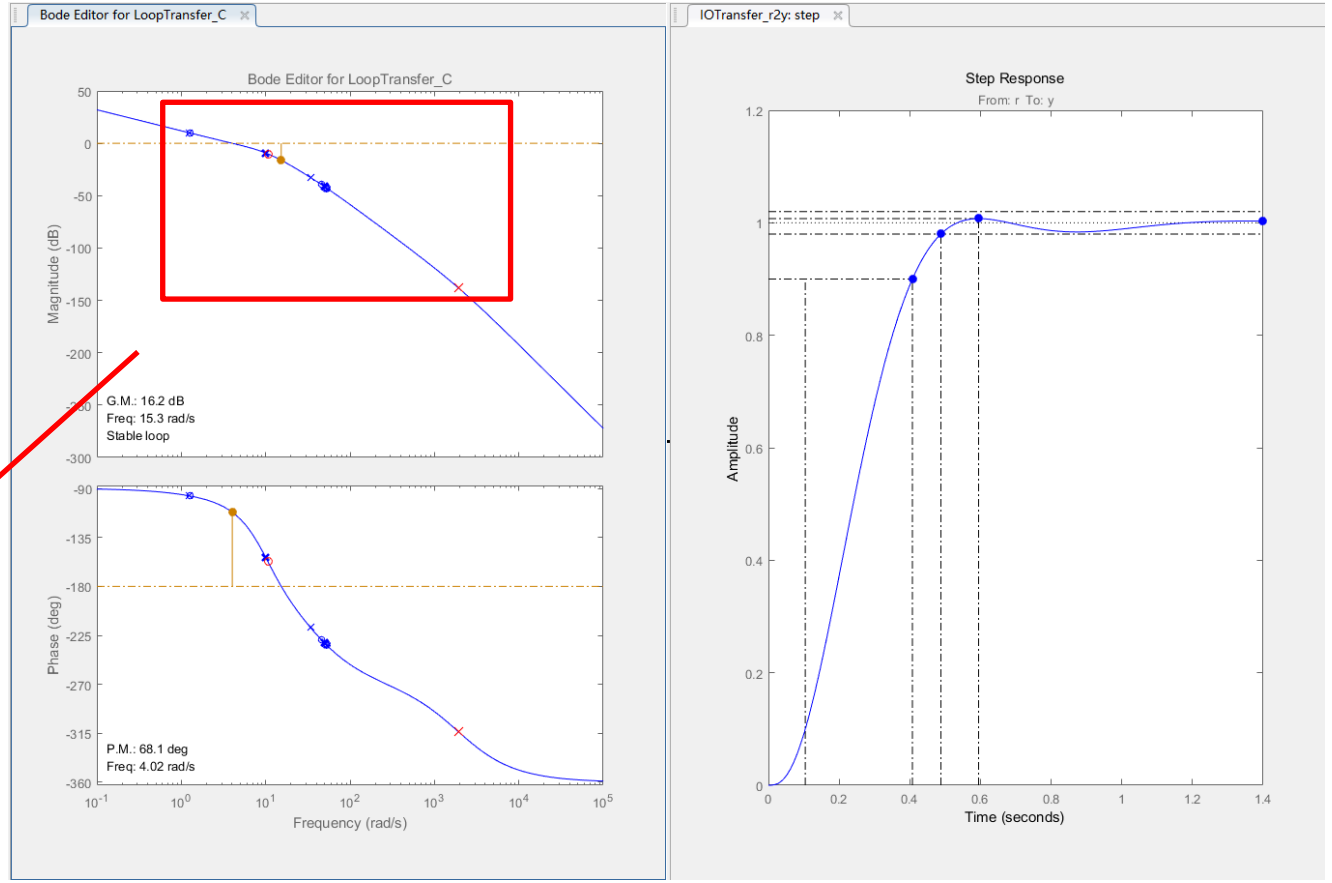


Figure. Step response after adding a lead compensator

Design Experiment

(5) Step5: Use toolbox to design the compensator

After this step, directly drag the zero and pole and observe their response to obtain an appropriate compensator.

$$G_c = \frac{4.0741(1+0.094s)}{1+0.01s}$$

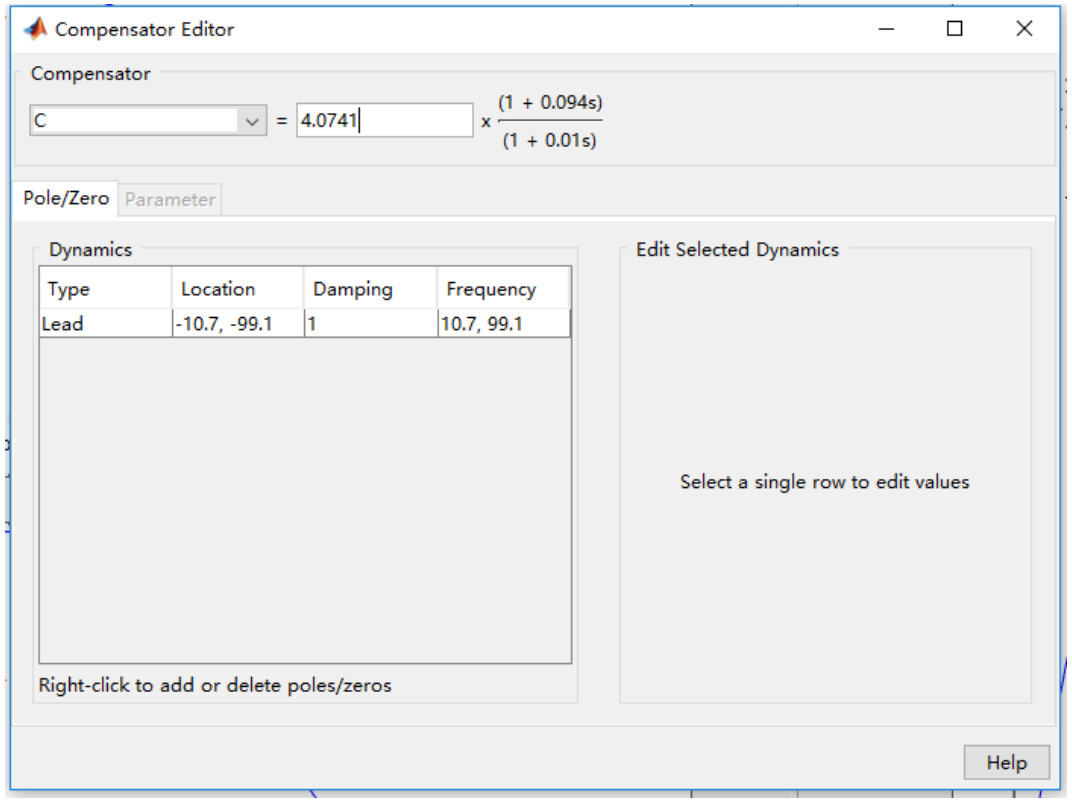


Figure: Compensator obtained using toolbox

Design Experiment

（6） Step6: Design a compensator for the position control loop of the x-axis channel

Put the velocity control loop compensator designed in Step 5 into the model in Step 1, The obtained Bode plot is shown below.

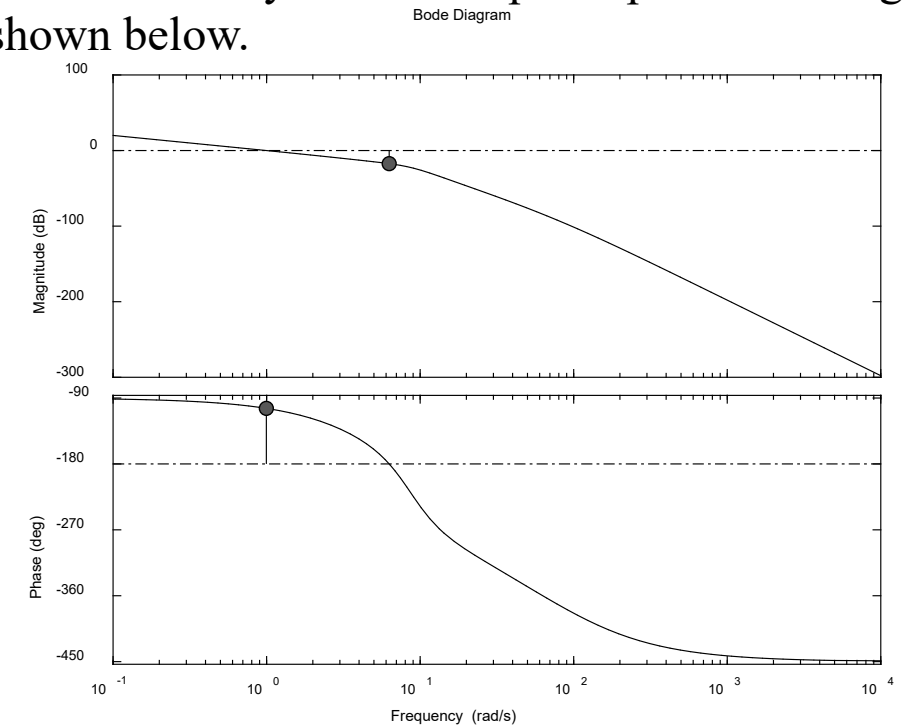


Figure: Bode plot of the position control loop

It can be observed that the phase margin is 75.8°, and the cut-off frequency is 0.99 rad/s. These meet the requirements marginally. Gain should be increased slightly to increase the cut-off frequency to 1 rad/s without violating the requirement that phase margin should be greater than 60°.

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= 0.01s”.

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

$$G_c = \frac{2.0452(1+0.15s)}{1+0.013s} \rightarrow G_c(z) = \frac{22.6\ z - 22.5}{z - 0.4634}$$

Design Experiment

☐ Simulation procedure

（2） Step2: Replace the control model

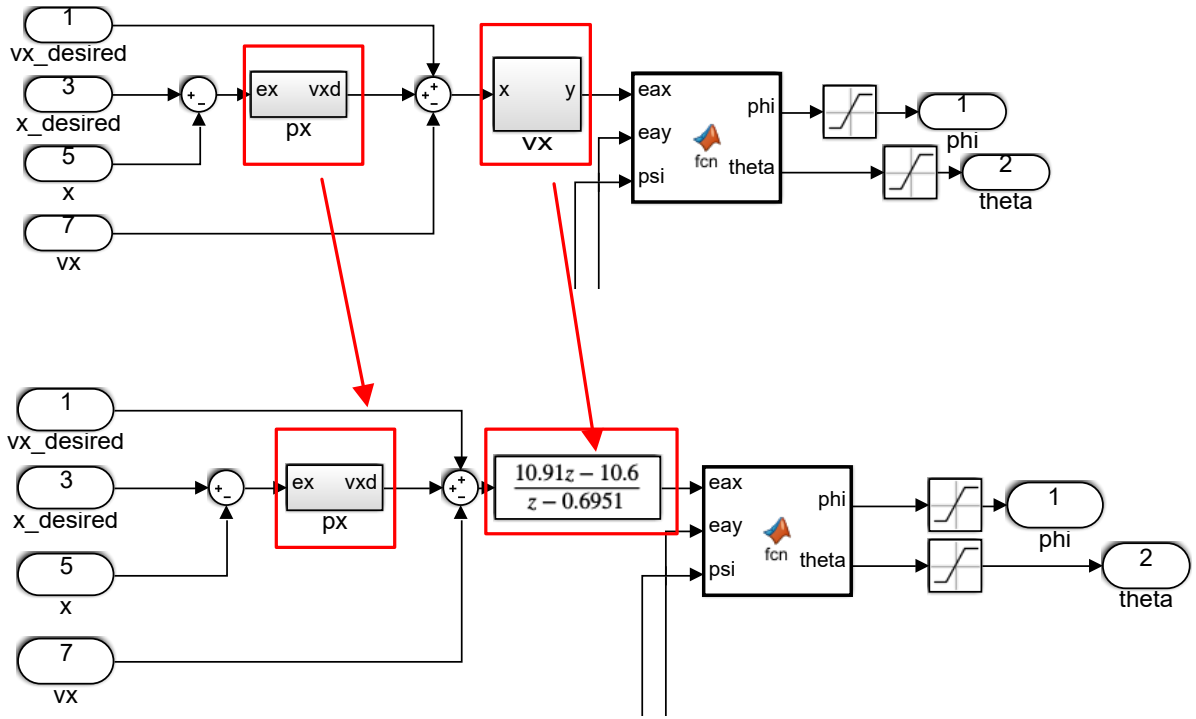


Figure: PID controller replaced in the HIL simulation model

Design Experiment

Simulation procedure

（3） Step3: HIL simulation

The quadcopter can fly along a straight line and hover.

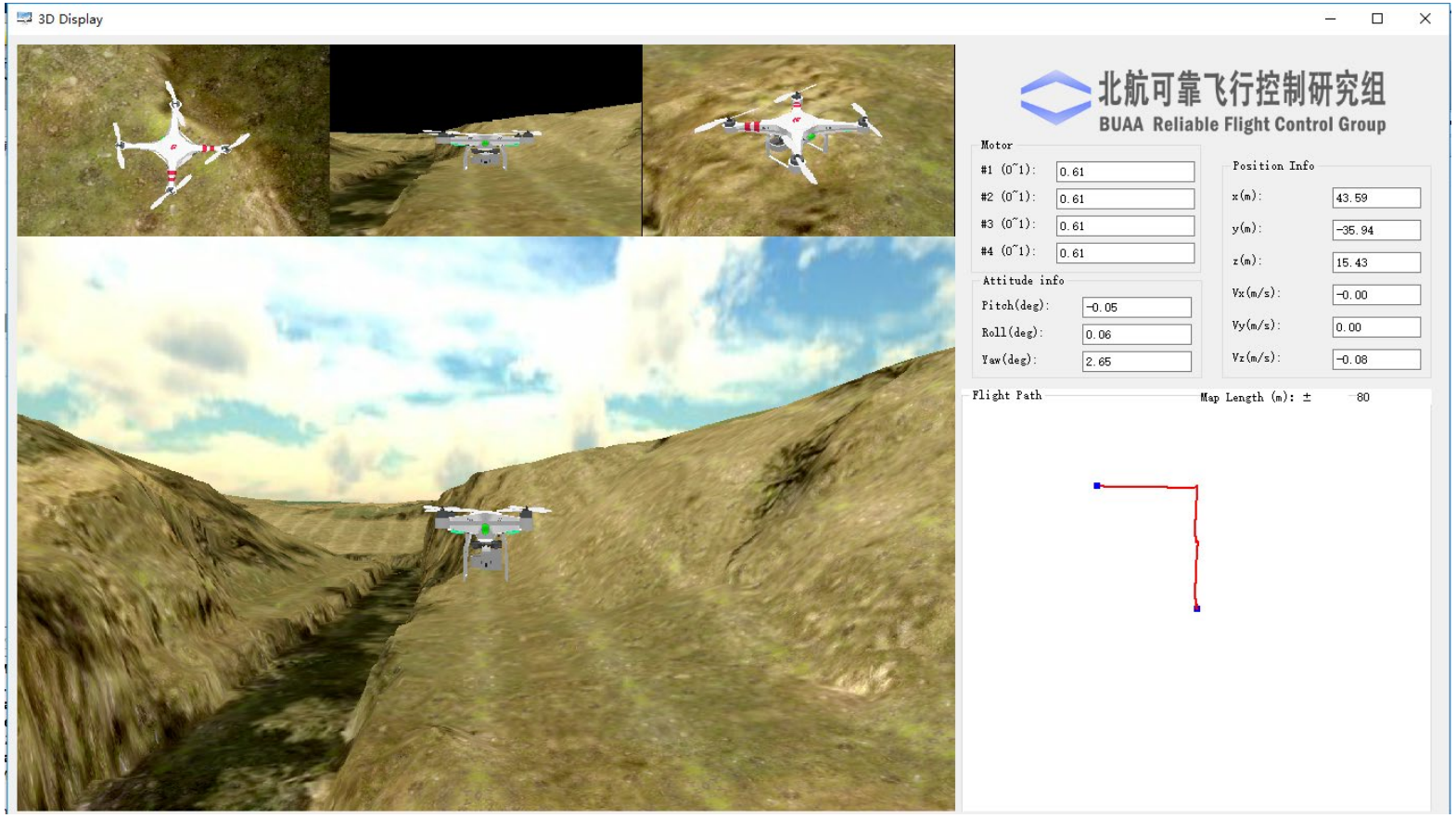


Figure: HIL simulation shown in 3DDisplay

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.

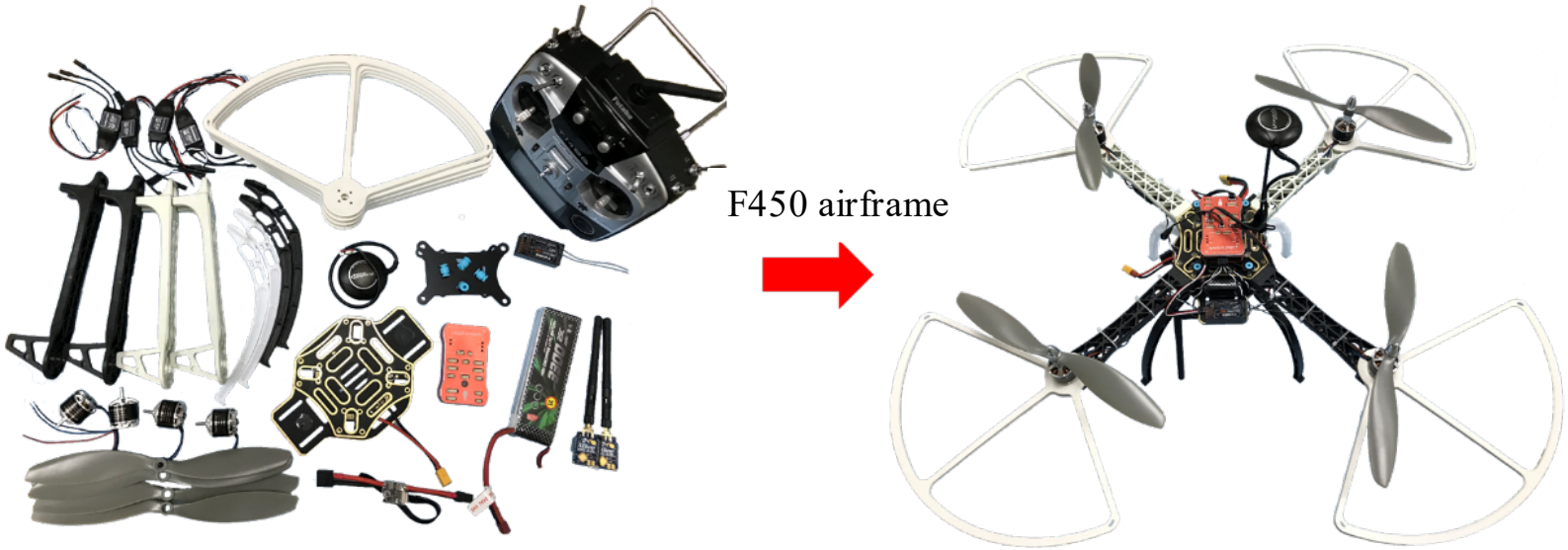


Figure: F450 airframe schematic

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. The detailed procedures of adding logger data can be found in Experiment 5.

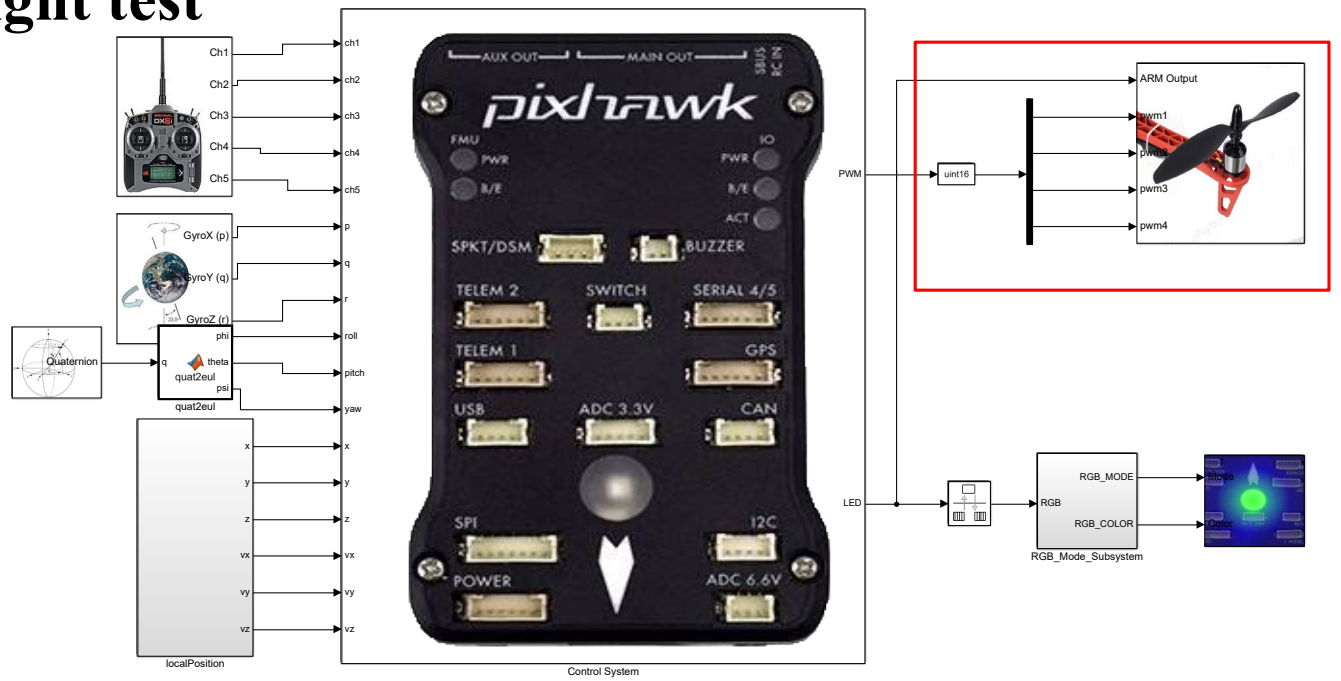


Figure: Model for flight test, Simulink model ” PosControl_FLY.slx”

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: Outdoor flight test

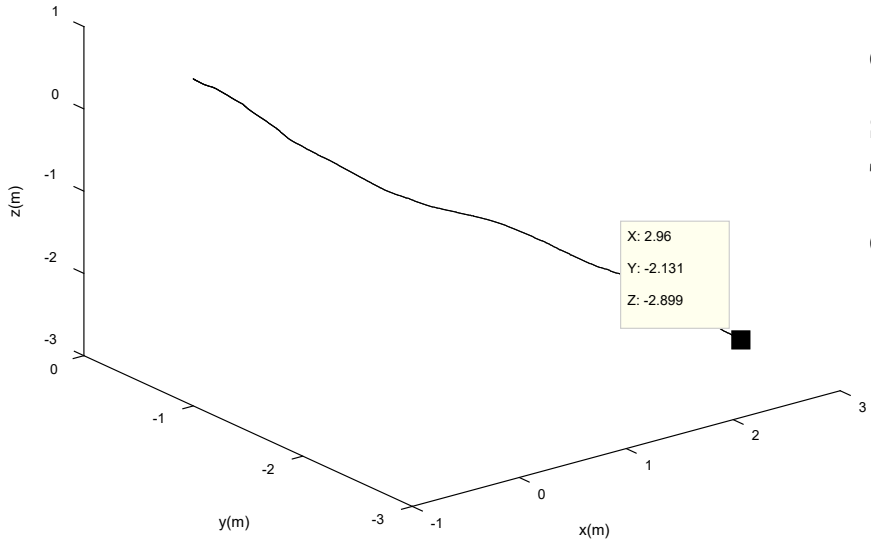
To ensure safety, a rope is tethered to the quadcopter, and the other end is tethered to a heavy object. The remote pilot maintains a safe distance from the quadcopter during flight.

Figure: Outdoor flight test

Design Experiment

☐ Flight Test Procedure

(5) Step5. Analyze the data



In the left-hand plot, the quadcopter reaches the specified position from 0. The designed set-point controller functions well.

The right-hand plot represents the velocity control response and it can be observed that the quadcopter flies at the specified speed with a fast velocity response

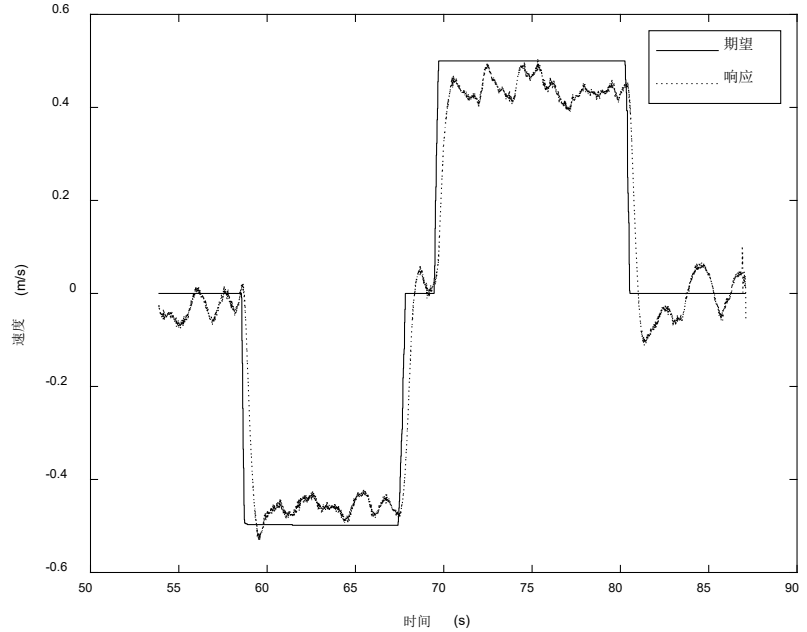


Figure: The position and velocity along the $o_e y_e$ axis

Summary

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

（2） The PSP tool of Simulink was used to generate the embedded code which was then uploaded to the Pixhawk autopilot for HIL simulation and flight test. .

（3） The parameters of the PID controller were adjusted to get the satisfied parameters. The system analysis tool in MATLAB/Simulink was adopted to obtain Bode plots corresponding to the open-loop position control system and velocity control system to observe the phase margin and gain margin of the corresponding closed-loop systems.

（4） In order to satisfy the given requirements, the system compensation method was adopted. Lead and lag-lead compensators were designed for the position control loop and velocity control loop respectively, which met the given requirements. Furthermore, the design was verified through HIL simulation and flight test.

Thanks