DELFT UNIVERSITY OF TECHNOLOGY

INTEGRATION PROJECT SYSTEMS AND CONTROL
SC42035

Integration Project: Helicopter

Authors:
Liuyi Zhu (5808383)
Qingyi Ren (5684803)

September 18, 2024

I. Introduction

In this experiment, the helicopter setup is used and analysed. This setup is composed of a beam attached to a fixed pole. The beam can freely rotate in the horizontal and vertical planes. At both ends of the beam, DC motors with propellers are attached. For simplicity, only the motor which is used to control the vertical angle (elevation) is set free and the motor which controls the horizontal angle (azimuth) is locked. The control object is to control the motors to achieve the desired reference. The schematic diagram is shown in Fig 1 and shows the system structure composed of the relevant variable. It is worth mentioning that positive directions of variables are determined by arrows.

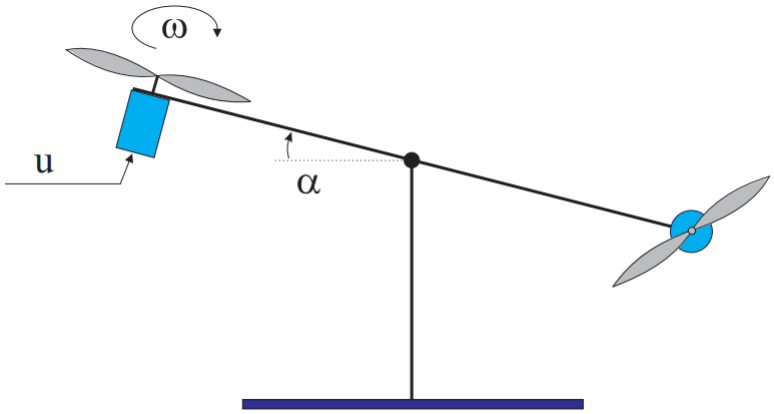


Fig. 1. Schematic drawing of the helicopter setup.

The input of the system is u which represents the voltage applied to the respective motor. This input needs to be scaled from -1 and 1. The -1 and 1 values of the input represent the maximal voltage making propeller and the beam rotate in the negative sense and maximal voltage making the propeller and the beam rotate in the positive sense respectively. The angular velocity of the propeller ω and the angle of the beam α are two outputs which could be measured by sensors. The measurement of ω is also needed to be scaled between -1 and 1. The following sections are to calibrate and scale the measurement, system identification to estimate the mathematical model or parameters of helicopter system based on input-output data and design the controllers based on LQR and MPC.

II. Calibration and Scaling

A. Calibration

In this section, the measurements coming from setup are checked if they could be trusted. In another word, the set up may have the biased measurements which could be eliminated by giving the system zero input and doing linear regression based on collected yaw sensor values and their corresponding physical values.

In the calibration test, the input u is set to zero and the α is fixed at 5 points $[-\pi/3, -\pi/6, 0, \pi/6, \pi/3]$, for the period of Simulink $T = 20$s, the measured α is collected during this period under five different values of actual α respectively and the mean value of the measured data under different actual α is calculated. By combining the actual α and its corresponding mean of the measured α to compose the data

points and using the MATLAB function **polyfit** of data points, the linear regression of the calibration is shown in Fig 2. The result of linear regression is shown as:

$$y = 1.023x - 0.0336$$

where y represents the Measured α and x represents True α. The slope of the linear line, which measures 1.023, closely aligns with the expected value of 1, providing strong evidence for the accurate calibration of the setup. To compensate the calibration error, the **Sum block** and **Constant block** are used to make zero point alignment. More specifically, the measured α should be added to 0.0336 to approximate the True α.

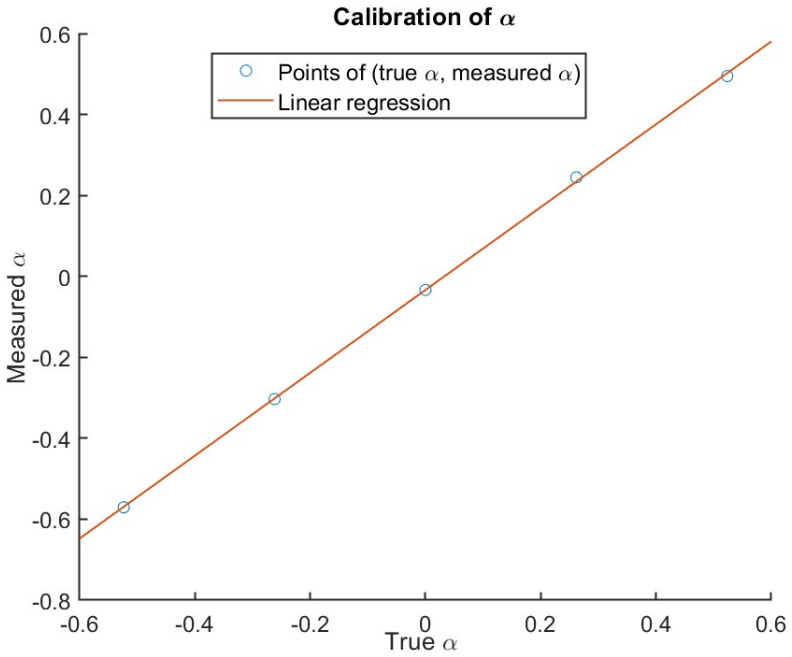


Fig. 2. The calibration of α.

B. Scaling

In this part, the measurement of helicopter ω is scaled between -1 and 1 where -1 corresponds to the maximal negative velocity and $+1$ corresponds to maximal negative velocity. In order to obtain the maximal and minimal velocity of ω, the input u is set at the minimal value -1 and maximal value $+1$ respectively to obtain the maximal absolute value of ω at around 370 rad/s. By dividing all measured ω by 370, the obtained ω is scaled between -1 and $+1$.

III. Physical Modelling

In this part, the physical modelling is constructed based on the Lagrange's Equations. The system of helicopter could be taken as the beam, the DC motor is taken as the outside part which provides the system with torque τ.

Considering the friction of beam, in the form of $f_{friction} = b\dot{\alpha}$ where $\dot{\alpha}$ represents the angular velocity of the beam and b is the friction coefficient during the rotation of the beam. The external force of the system is defined as:

$$F = \tau - b\dot{\alpha} \tag{1}$$

The Lagrange Equations are built to depict the nonlinear system model as:

$$\begin{aligned} L &= T - U \\ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} &= F \end{aligned} \tag{2}$$

where T and U represent kinetic energy and potential energy respectively. The kinetic energy is the rotational energy of the beam and could be written as:

$$T = \frac{1}{2}J\dot{\alpha}^2 \tag{3}$$

where J represents the moment of inertia of the entire rotating component, which includes the propellers on both sides, the beam, and the weight suspended below.
The potential energy of the system is the gravitational energy. When the beam rotates, the propellers and beams on both sides rise and fall. The mass and displacement on both sides are equal, thus canceling out the changes in gravitational potential energy. Therefore, the change in gravitational potential energy of the system is equivalent to the change in gravitational potential energy of the hanged heavy object.
When the rotation angle of the beam isα, the change in height of the center of gravity of the heavy object is $l(1 - cos(\alpha))$, where l is the length of the lever connecting the heavy object. The potential energy of system can be represented as:

$$U = mgl(1 - cos(\alpha)) \tag{4}$$

where m represents the mass of lever and heavy object, and g is gravity acceleration. Combining the equations (1), (2), (9) and (4), the Lagrange equations fo this system could be written as:

$$L = T - U = \frac{1}{2}J\dot{\alpha}^2 - mgl(1 - cos(\alpha)) \tag{5}$$

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} = J\ddot{\alpha} + mglsin(\alpha) = \tau - b\dot{\alpha} \\ &\Rightarrow J\ddot{\alpha} + b\dot{\alpha} + mglsin\alpha = \tau \end{aligned} \tag{6}$$

In a propeller system, the magnitude of the thrust generated by the propeller is approximately proportional to the square of the propeller's rotational speed. [1] Furthermore, the product of thrust and the length of the beam is equal to the torque generated by the propeller. When the propeller rotates in the opposite direction, it generates thrust in the opposite direction as well. Therefore, the relationship between torque and propeller rotational speed can be expressed as:

$$\tau = k|\omega|\omega \tag{7}$$

where k is unknown coefficient.
Also the input of the system is the voltage applied to DC motor. Thus the next step is to find the relation of u to ω, which could relate the input u to torque τ. The relationship between u and τ can also be derived by Lagrange Equation:

$$\begin{aligned} L_1 &= T_1 - U_1 \\ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L_1}{\partial \dot{\omega}} - \frac{\partial L_1}{\partial \omega} &= F_1 \end{aligned} \tag{8}$$

where T_1 and U_1 are kinetic energy and potential energy of propeller system respectively. It is obvious that the potential energy of propeller system is zero and the kinetic energy is:

$$T_1 = \frac{1}{2} J_1 \omega^2 \tag{9}$$

where J_1 is the moment of inertia of propeller.

$$L_1 = T_1 - U_1 = \frac{1}{2} J_1 \omega^2 \tag{10}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L_1}{\partial \dot{\omega}} - \frac{\partial L_1}{\partial \omega} = -J_1 \omega \tag{11}$$

The external forces F_1 acting on the propeller system include the torque transmitted from the DC motor to the propeller, the air resistance generated by the propeller rotation, and the frictional force on the propeller rotation axis.
The frictional force can represent as $k_1\dot{\omega}$, where k_1 is the friction coefficient during the rotation of propeller. Then expression of air resistance is similar with the thrust generated by the propeller, which is proportional to the square of propeller's rotational speed:

$$f_{air} = k_2 \omega^2 \tag{12}$$

The torque generated by DC motor is directly proportional to the current flowing through the motor:

$$\tau_1 = k_3 I \tag{13}$$

In this system, the operating frequency of the motor is relatively low, the effect of inductance on the current-voltage relationship is negligible. Therefore, the influence of inductance can be disregarded. The voltage of the motor can be represented as the product of the current and the resistance:

$$u = rI \tag{14}$$

Combine (8) to (14), we can get the relationship between input u and state ω:

$$-J_1 \omega = \frac{k_3}{r} u - k_1 \dot{\omega} - k_2 \omega^2 \tag{15}$$

$$k_1 \dot{\omega} = -k_2 \omega^2 + J_1 \omega + \frac{k_3}{r} u \tag{16}$$

Defining the state vector $x = [\omega, \alpha, \dot{\alpha}]^T$, the nonlinear state space model could be depicted as:

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \omega \\ \alpha \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} \dot{\omega} \\ \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = f(x,u) = \begin{bmatrix} f_1(x,u) \\ f_2(x,u) \\ f_3(x,u) \end{bmatrix} = \begin{bmatrix} -\frac{k_2}{k_1}\omega^2 + \frac{J_1}{k_1}\omega + \frac{k_3}{rk_1}u \\ \dot{\alpha} \\ -\frac{b}{J}\dot{\alpha} - \frac{mgl}{J} sin(\alpha) + \frac{k}{J}|\omega|\omega \end{bmatrix} \tag{17}$$

To simplify the system function, we replace all unknown values in (17) by p:

$$\dot{x} = \begin{bmatrix} \dot{\omega} \\ \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} p_1\omega + p_2\omega^2 + p_3 u \\ \dot{\alpha} \\ p_4\dot{\alpha} + p_5 sin\alpha + p_6|\omega|\omega \end{bmatrix} \tag{18}$$

Around the equilibrium point $x = x_e$ and $u = u_e$ which make $f(x_e, u_e) = 0$, the linearized state space model could be written as:

$$\begin{aligned} A &= \begin{bmatrix} \frac{\partial f_1(x,u)}{\partial x_1} & \frac{\partial f_1(x,u)}{\partial x_2} & \frac{\partial f_1(x,u)}{\partial x_3} \\ \frac{\partial f_2(x,u)}{\partial x_1} & \frac{\partial f_2(x,u)}{\partial x_2} & \frac{\partial f_2(x,u)}{\partial x_3} \\ \frac{\partial f_3(x,u)}{\partial x_1} & \frac{\partial f_3(x,u)}{\partial x_2} & \frac{\partial f_3(x,u)}{\partial x_3} \end{bmatrix} \\ &= \begin{bmatrix} p_1 + 2p_2\omega & 0 & 0 \\ 0 & 0 & 1 \\ 2p_6|\omega| & p_5 cos\alpha & p_4 \end{bmatrix} \Bigg|_{x=x_e} \end{aligned} \tag{19}$$

$$B = \begin{bmatrix} \frac{\partial f_1(x,u)}{\partial u} \\ \frac{\partial f_2(x,u)}{\partial u} \\ \frac{\partial f_3(x,u)}{\partial u} \end{bmatrix} = \begin{bmatrix} p_3 \\ 0 \\ 0 \end{bmatrix} \Bigg|_{u=u_e} \tag{20}$$

$$\dot{x} = \begin{bmatrix} p_1 + 2k_2\omega & 0 & 0 \\ 0 & 0 & 1 \\ 2p_6|\omega| & p_5 cos\alpha & p_4 \end{bmatrix} \Bigg|_{x=x_e} x + \begin{bmatrix} p_3 \\ 0 \\ 0 \end{bmatrix} u \tag{21}$$

By employing the Forward Euler discretisation method, the continuous-time dynamic model (21) could be further transformed to the discrete-time dynamic model:

$$x(k+1) = A_d x(k) + B_d u(k) \tag{22}$$

where $A_d = I + AT$, $B_d = BT$, I is 3×3 identity matrix, and T is the sampling time.

IV. System identification

Based on the physics model built in the previous section,the state space model of helicopter system could be written as:

$$\dot{x} = \begin{bmatrix} \dot{\omega} \\ \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} p_1\omega + p_2\omega^2 + p_3 u \\ \dot{\alpha} \\ p_4\dot{\alpha} + p_5 sin\alpha + p_6|\omega|\omega \end{bmatrix} \tag{23}$$

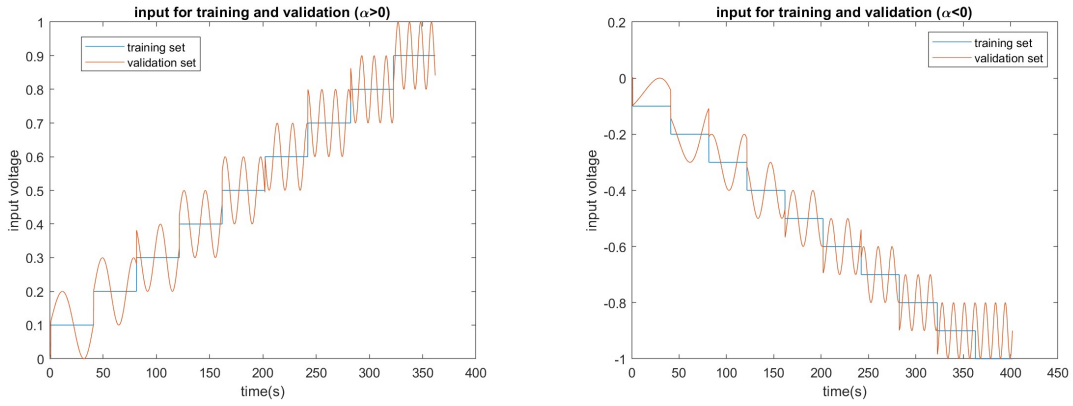
where p_1, p_2, p_3, p_4 , p_5 and p_6 are the unknown variables.

We have derived the physical model of the system, with only a few coefficients remaining unknown. Furthermore, this is a nonlinear system. Therefore, we have decided to use the *idnlgrey* function in MATLAB for system identification.

After multiple attempts, we found it challenging to identify a single set of parameters that can be applied to both cases of $(\alpha > 0)$ and $(\alpha < 0)$ in the system. Therefore, we have decided to divide overall system into two parts, one for positive α and the other for negative α, and perform system identification separately for each part.

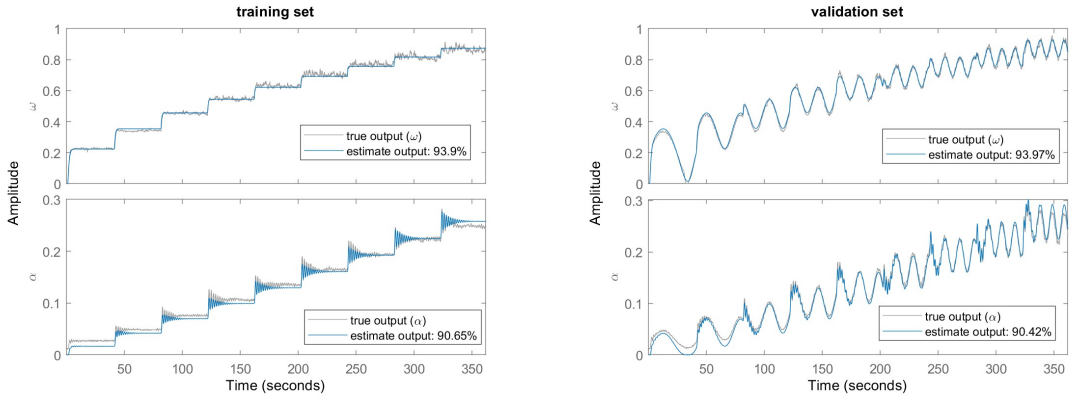
For each part, we generated two input signals separately and fed them into the model. One signal was used for training to obtain estimated values of the unknown coefficients, while the other signal was used to validate the accuracy of the model. The train and validation signal of two parts are shown in Figure 3. From the figure, it can be observed that the range of the generated input voltage almost covers the entire interval from -1 to 1. Therefore, we have obtained comprehensive model information during the system identification process. This approach ensures that the identified model is more accurate and applicable to the entire input range.

The identification results of two parts are shown in Figure 4 and 5. For both the training set and the validation set, the accuracy of the estimated output compared to the true output of the system is



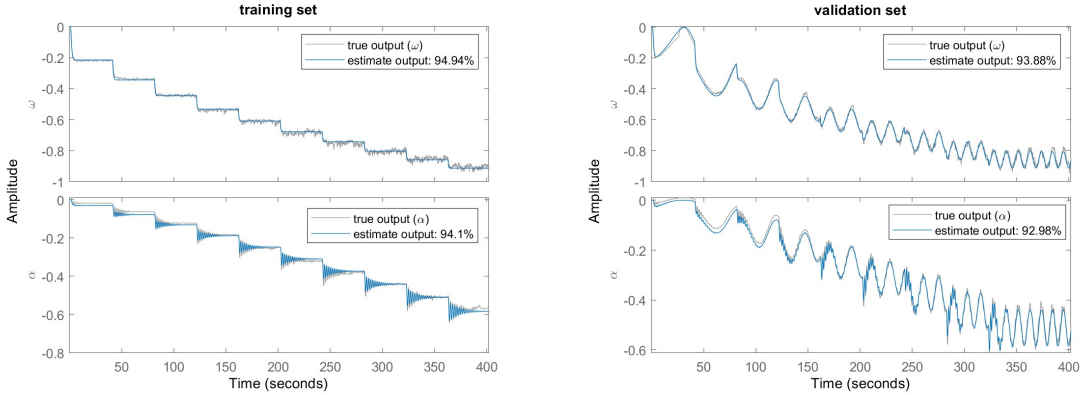
(a) $\alpha > 0$ (b) $\alpha < 0$

Fig. 3. input for training and validation



(a) training set (b) validation set

Fig. 4. identification result($\alpha > 0$)



(a) training set (b) validation set

Fig. 5. identification result($\alpha < 0$)

above 90%. This indicates that the results obtained from the system identification process can effectively represent the true model.
The identification results are shown below:

1) $\alpha > 0:\ p_1 = -0.4229, p_2 = -1.5532, p_3 = 1.7371, p_4 = -0.2499, p_5 = -11.5731, p_6 = 3.8587$
2) $\alpha < 0:\ p_1 = -0.5089, p_2 = 1.6877, p_3 = 1.8696, p_4 = -0.1981, p_5 = -11.1035, p_6 = -7.3444$

V. Observer design

The Luenberger observer is a method to estimate the state variables of a system based on its inputs and outputs. In this project, the states ω and α are measurable but the state $\dot{\alpha}$ is not directly measurable. To estimate $\dot{\alpha}$, the Luenberger observer is used by employing dynamic model of the system and the available measurements of ω and α under different values of input u to estimate internal system's states.

Based on the systems dynamics $\dot{x} = Ax + Bu$ and output measurement $y = Cx + Du$, the mathematical form of the Luenberger observer is given as:

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - \hat{y}(k)) \tag{24}$$

$$\hat{y}(k) = C\hat{x}(k) \tag{25}$$

where $\hat{x}$ is the estimated state vector of the system and $\hat{y}$ is estimate output.

L is the observer gain matrix which is typically chosen such that it stabilizes the observer dynamics and provides accurate state estimation. In this assignment, the pole placement method is used to choose the suitable L and the Luenberger observer block in Simulink is shown in Fig (6).

In MATLAB, the function **place** function is used to design optimum gain matrix L which can make the poles of the system achieving at desirable place, thereby ensuring system stability [3]. In this assignment, we have chosen three poles at 0.3, 0.25, and 0.2 respectively. Firstly, all three poles are located inside the unit circle, which ensures the stability of the system with observer. Additionally, we aim to observe the system's state in time, in order to achieve faster control. Hence, we have selected these three poles that are relatively close to the origin, which results in a faster convergence speed of the observer. However, poles that are closer to the origin are also more sensitive to changes in the true state of the system, meaning they can amplify disturbances and measurement noises to some extent.

After testing several sets of different pole selections, we ultimately chose poles at 0.3, 0.25, and 0.2. This set of poles allows the observer to track the system state promptly without amplifying the noise.

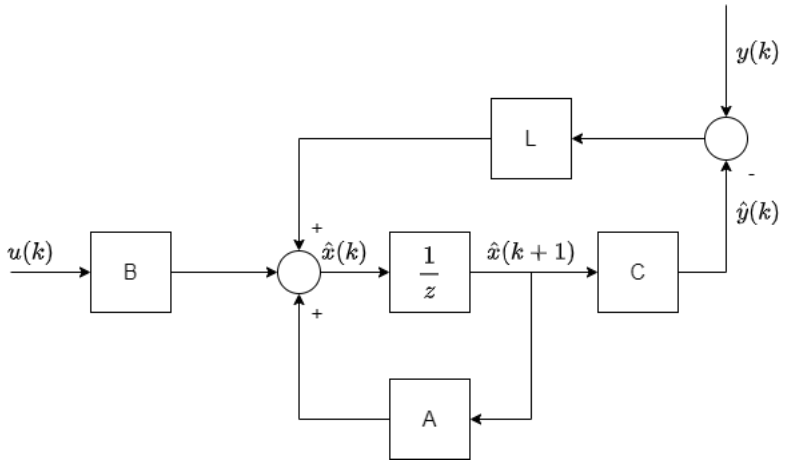


Fig. 6. The structure of Luenberger observer under the case of stabilizing at one equilibrium.

The Luenberger observer is primarily used for observing linear systems. When the helicopter system is engaged in the reference tracking task with varying α values, the equilibrium points undergo changes. In this case, a fixed-gain observer such as the Luenberger observer is no longer applicable.

Computing the linear system at each α point to design the Luenberger observer gain would introduce significant computational overhead. Additionally, dividing the operating region of α into sub-regions and designing the corresponding Luenberger observer gain in each sub-region can lead to unsmooth transitions when switching between these regions. This lack of smoothness arises because the Luenberger observer gain may not be continuous, potentially resulting in instability during the switching process.

To mitigate the computational effort and enhance the smoothness of transitions between different α sub-regions, a look-up table approach is employed. This involves designing the table based on several carefully chosen α points and their corresponding Luenberger observer gains. By precomputing and storing these gains in the **look-up table dynamic**, the need for on-the-fly computation of observer gains for each α value is eliminated, thereby reducing the computational burden. Furthermore, the use of the **look-up table dynamic** promotes smoother transitions between sub-regions of α, ensuring stability during the switching process.

As the reference tracking problem is based on α, the turning points are also set based on different α. The input u is set to take values from the set

$$[-1, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$$

and the equilibrium and its corresponding Luenberger gain L is calculated accordingly under different control u.

It is worth mentioning that there are two pairs of system identification parameters: parameters when $\alpha > 0$ and parameters when $\alpha \leq 0$. When the reference of α is larger than 0, the first pair of system parameters are used to calculate the equilibrium and its corresponding linearized system. Furthermore, instead of using the linear structure of observer to obtain $\hat{x}(k+1)$, the nonlinear system dynamics is used to estimate $\hat{x}(k+1)$, that is:

$$\hat{x}(k+1) = f(\hat{x}(k), u(k)) = \begin{bmatrix} \hat{x}_1(k) + T(p_1\omega + p_2\omega^2) \\ \hat{x}_2(k) + T\hat{x}_3(k) \\ \hat{x}_3(k) + T(p_4\hat{x}_3(k) + p_5 sin(\hat{x}_2(k) + p_6|\hat{x}_1(k)|\hat{x}_1(k)) \end{bmatrix} + \begin{bmatrix} Tp_3 \\ 0 \\ 0 \end{bmatrix} u(k)$$

with higher accuracy shown in Fig (7). The Luenberger gain is a 3×2 matrix and its 6 elements are used as the table points corresponding to the turning points of α.

The performance of the observer is illustrated in Figure 8. From the figure, it can be observed that the observer we designed is capable of accurately estimating the system's state, as the estimated output closely aligns with the true output of the system.

VI. Controller design

Firstly, we need to determine the sampling time (T) of the system. A too short sampling time would increase computational load, while a too long sampling time would decrease control accuracy. After several attempts, we have decided to set the sampling time to 0.05 seconds.

In this section, we have designed two types of controllers: LQR (Linear Quadratic Regulator) and MPC (Model Predictive Control). Both controllers have been utilized to achieve two control tasks: stabilization at a specific point and reference tracking.

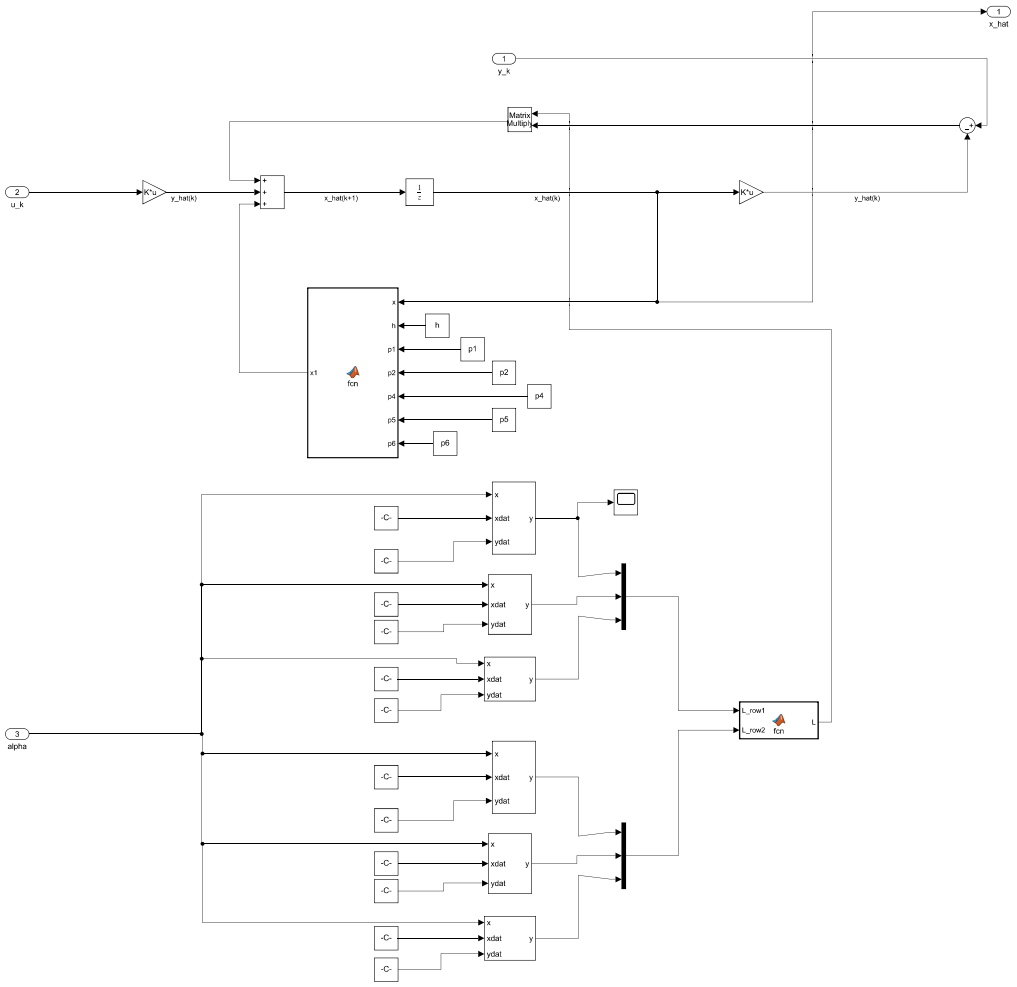


Fig. 7. The block of nonlinear Luenberger observer under the case of reference tracking.

A. *LQR controller*

LQR controller is the method to construct quadratic optimization problem to obtain the control policy. This optimization trade off tracking desired states and minimizing control effort. By using weighting matrices Q and R which reflect the penalties associated with different states and inputs, the cost function is defined to quantify the system's performance and typically includes a combination of the system's state variables and control effort. The goal is to minimize this cost function and represented as:

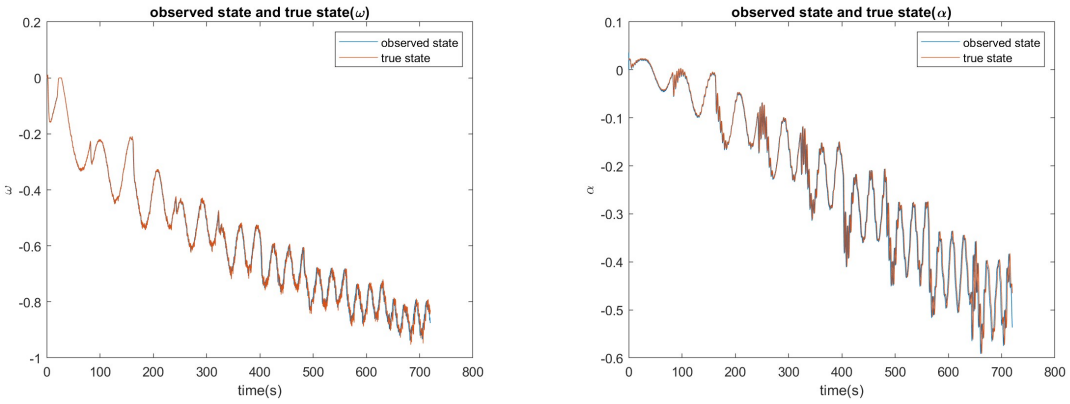
$$\min \int_{t_0}^{\infty} (u(t)^T R u(t) + x(t)^T Q x(t)) dt \tag{26}$$

where $Q \in R^{3\times 3}$ is a positive definite weighting matrix that determines the penalty on the system's state variables and $R \in R^{+}$ is a positive semi-definite weighting matrix that defines the penalty on the control effort. Then the optimal control gains are obtained by

$$K = -(R + B^T P_n B)^{-1} B^T P_n A \tag{27}$$

where P_n represents solution to algebraic Riccati equation:

$$P_n = A^T P_n A - A^T P_n B (R + B^T P_n B)^{-1} B^T P_n A + Q$$



(a) ω (b) α

Fig. 8. comparison of estimated state and true state

Then the optimal control input could be represented as:

$$u(t) = -Kx(t) \tag{28}$$

1) Stabilizing at one equilibrium: In this scenario, we begin by selecting a desired alpha value. Using this fixed α value, we can compute the corresponding equilibrium and linearize the nonlinear system around this equilibrium.

Since the system's motion predominantly occurs in the proximity of this equilibrium point (except for a brief period when jumping from $\alpha = 0$ to the desired α), we can utilize the linearized model at this point to effectively control the system.

After selecting appropriate Q and R matrices, we use **lqr** function in MATLAB to compute the controller gain K. In the following sections, we will fine-tune various parameters in Q and R matrices and analyze their impact on system control. Consequently, the control input of the system is obtained by multiplying K with Δx (the difference between the true state and the equilibrium point state). The block diagram of controlled system is depicted in Figure 9.

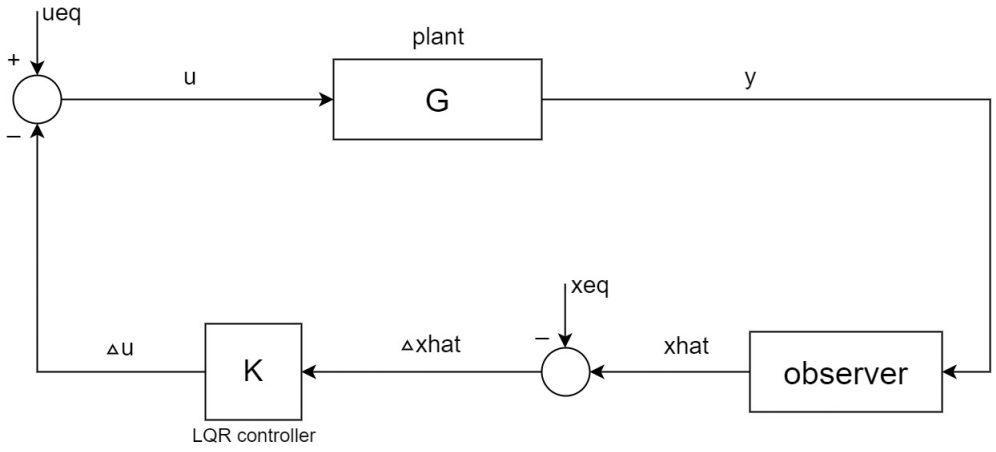


Fig. 9. The structure of LQR controller under the case of stabilizing at one equilibrium.

2) Reference tracking: In the case of reference tracking, there is significant variation in α. Therefore, the same linear model around one equilibrium is no longer applicable with α having large variation. This implies that the controller gain matrix K needs to vary based one different regions of α.

Similar to the observer design, we have computed multiple equilibrium points based on different input values. Subsequently, the linearized systems around each equilibrium points is obtained. With different pairs of A and B matrices of different linearized systems, the corresponding controller gain K could be calculated by **lqr** function.

The input u is set to take values from the set

$$[-1, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$$

and then the equilibrium and its corresponding LQR gain K is calculated accordingly under different input u. The LQR gain is a 1×3 matrix and its 3 elements are used as the table points corresponding to the turning points α which is shown in Fig (10).

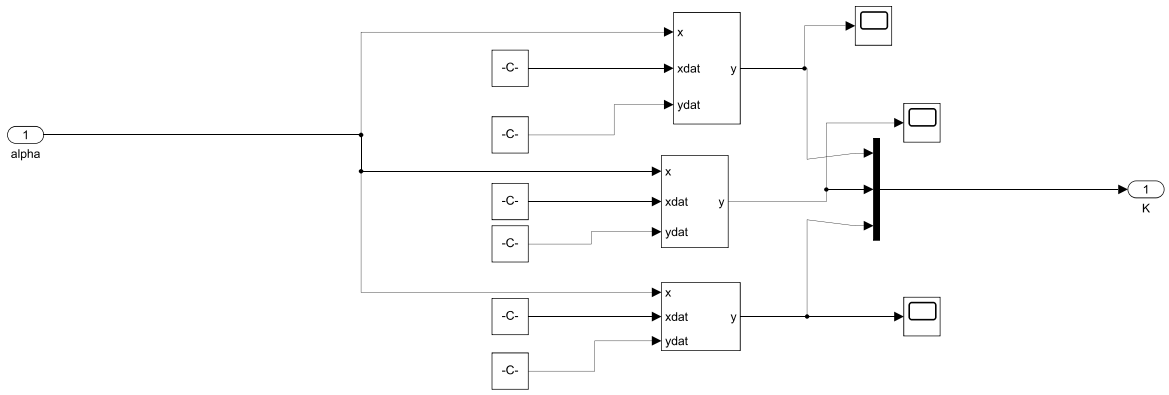


Fig. 10. Look-up table dynamic of optimal control gain K in Simulink.

It is crucial to emphasize that the α used in the look-up table to obtain the interpolated value is not the system's actual α, but rather the desired reference α specific to that particular time point. This distinction is important because our objective is to minimize the difference between the actual α and the reference value of α. To achieve this, we leverage the system dynamics in the vicinity of the reference point to calculate the controller gain and regulate the system accordingly. By adopting this approach, we enhance the system's ability to accurately follow the reference signal, resulting in improved tracking of the true

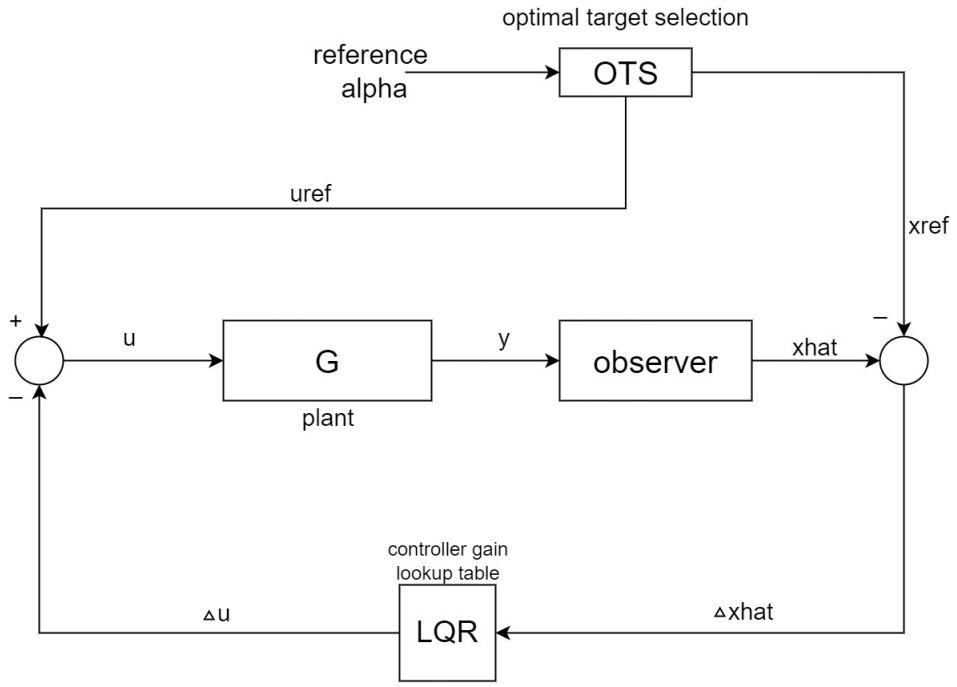


Fig. 11. The structure of LQR controller under the case of reference tracking.

α. The block diagram of controlled system is depicted in Figure 11.

3) tune Q and R: The control objective of stabilizing at one equilibrium is to achieve stability at the desired equilibrium point, where the system can reach a steady state without any offset. Additionally, we aim to minimize overshoot and reach steady state as soon as possible.

To optimize reference tracking, our control objective is to minimize the disparity between the actual α of the system and the desired reference value. Simultaneously, we aim to achieve a swift and precise performance during the tracking process, while ensuring that no steady error persists.

By adjusting the elements of the weighting matrices Q and R, we can tune the performance and behavior of the controller and achieve the desired control effect.

Assume

$$Q = diag[q_1 \quad q_2 \quad q_3],\ R = r \tag{29}$$

First, we conducted experiments with different values of q_2 and r. Among the four adjustable parameters, these two parameters are relatively more important. Since our desired control performance is to stabilize α at a specific value, the weight corresponding to α, which is q_2 representing the state cost of α, is crucial to overall state cost.

The selection of q_2 directly influences the importance placed on accurately tracking and controlling α during the system's operation. By tuning the value of q_2, we can effectively adjust the relative importance of α in the state cost. More specifically increasing the weighting factor amplifies the significance of α, leading to a stronger emphasis on achieving precise control over this state. Conversely, decreasing the value of q_2 reduces the relative importance of α in the state cost, allowing other states or control objectives to receive more attention.

On the other hand, r represents the weight of the inputs and has a significant impact on the control performance of the system.

We experimented with different value of $\frac{q_2}{r}$ to observe the system's performance. When this ratio is relatively large, indicating that the weight on the system state is greater, the controller becomes more proactive in reducing the error in α. In this scenario, the system can reach the steady state faster, and the overshoot is slightly reduced. This is because the larger value of q_2 ensures that the system avoids having a significant deviation from the desired steady value for α. However, after the system reaches stability, if there is a small disturbance that causes α to deviate from the steady value, the system will try to return to the steady value quickly. This can result in small oscillations after reaching stability, preventing the system from settling at the equilibrium point.

When the ratio is relatively small, indicating a higher weight on the inputs. The controller will become more conservative, resulting in a slower response speed. Correspondingly, when the system reaches the steady state, it can remain stable at the steady value. If a disturbance acts on the system, it will smoothly return to the steady state without generating oscillations or significant deviations.

When q_2/r is set to 20, the system exhibits favorable performance in various aspects. It can stabilize at the steady state, while keeping the settling time and overshoot relatively small. Afterward, we adjusted q_1 and q_3 to further improve the control performance.

By choosing an appropriate value for q_1, we can reduce control overshoot while increasing the system's stability. Since the first state of the system represents the angular velocity of propeller, a larger value of q_1 can prevent drastic changes in angular velocity, thereby reducing the likelihood of system oscillations. However, if the value of q_1 is too large, it can lead to an increase in overshoot and prolong the settling time.

If the value of q_3 is appropriately increased, it can lead to a reduction in overshoot and an improvement in system stability. However, if the value of q_3 becomes excessively large, the system's settling time will significantly increase, and it will take more time for the system to return to the steady state in the presence of disturbances. This is because the third state of the system represents the rate of change of α, and its steady value is always zero. If the corresponding weight is too large, the system tends to maintain its current state $\ddot{\alpha}$ (minimizing the rate of change of α) instead of actively controlling the system to return to the steady state when it deviates from it.

It is important to find the right balance and choose an appropriate weight for states and input to achieve the desired control objectives while ensuring system stability and a timely response to disturbances. Finally, we selected the following parameters:

$$Q = diag[15 \quad 250 \quad 5],\ R = 10 \tag{30}$$

These parameters are also applicable for reference tracking. The system can promptly track the reference and exhibit smooth motion.

B. MPC controller

MPC is a control strategy that utilizes a dynamic model of the system to make predictions and optimize control actions over a finite time horizon. By designing an appropriate cost function, we can control the system by obtaining the control input u that minimizes the cost function within the prediction horizon.

1) Stabilizing at one equilibrium: In this case, the cost function is designed to minimize the Δx (the difference between the true state and the equilibrium point state) and Δu (the difference between true

input and corresponding input at this equilibrium point):

$$\begin{aligned} J(k) &= \sum_{i=k}^{i=k+N_p-1} l(\Delta\hat{x}(i), \Delta u(i)) \\ &= \sum_{i=k}^{i=k+N_p-1} (\hat{x} - x_{eq})^T Q(\hat{x} - x_{eq}) + (u - u_{eq})^T R(u - u_{eq}) \\ &= \sum_{i=k}^{i=k+N_p 1} \Delta\hat{x}^T Q \Delta\hat{x} + \Delta u^T R \Delta u \end{aligned} \tag{31}$$

where Q and R are the weighting matrices for state variables and input variables respectively. They are used to quantify the importance of each state and input in the control process. N_p is the prediction horizon. x_{eq} is the equilibrium point state and u_{eq} is the corresponding input at this equilibrium point.

$\Delta\mathbf{x}$ and $\Delta\mathbf{u}$ represent the data vector of values of $\Delta y(k)$ and $\Delta u(k)$ up to the time step $k + N_p - 1$:

$$\Delta\mathbf{x} = \begin{bmatrix} \hat{x}(k) - x_{eq} \\ \hat{x}(k+1) - x_{eq} \\ \vdots \\ \hat{x}(k+N_p-1) - x_{eq} \end{bmatrix} \quad \Delta\mathbf{u} = \begin{bmatrix} u(k) - u_{eq} \\ u(k+1) - u_{eq} \\ \vdots \\ u(k+N_p-1) - u_{eq} \end{bmatrix}$$

To represent this cost function in MATLAB, we performed the following calculations:

Firstly, we linearize the nonlinear system around the equilibrium and get the system matrix A and B. Then $\Delta x(k+1)$ ... ,$\Delta x(k+N_p-1)$ can be represented as:

$$\begin{aligned} \Delta\hat{x}(k+1) &= A\Delta\hat{x}(k) + B\Delta u(k) \\ \Delta\hat{x}(k+2) &= A^2\Delta\hat{x}(k) + AB\Delta u(k) + B\Delta u(k+1) \\ &\vdots \\ \Delta\hat{x}(k+Np-1) &= A^{N_p-1}\Delta\hat{x}(k) + A^{N_p-2}B\Delta u(k) + ...B\Delta u(k+N_p-1) \end{aligned} \tag{32}$$

This can also be represented in matrix form as follows:

$$\begin{bmatrix} \Delta\hat{x}(k) \\ \Delta\hat{x}(k+1) \\ \vdots \\ \Delta\hat{x}(k+N_p-1) \end{bmatrix} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^{N_p-1} \end{bmatrix}}_{S} \Delta\hat{x}(k) + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B & A^{N_p-3}B & \dots & 0 \end{bmatrix}}_{T} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix} \tag{33}$$

$$\Delta\mathbf{x} = S\Delta x(k) + T\Delta\mathbf{u} \tag{34}$$

Assume

$$\bar{Q} = \begin{bmatrix} Q & & & & \\ & Q & & & \\ & & Q & & \\ & & & \ddots & \\ & & & & Q \end{bmatrix}, \bar{R} = \begin{bmatrix} R & & & & \\ & R & & & \\ & & R & & \\ & & & \ddots & \\ & & & & R \end{bmatrix} \tag{35}$$

Then the cost function can be represented as:

$$J(k) = \Delta\mathbf{x}^T \bar{Q} \Delta\mathbf{x} + \Delta\mathbf{u}^T \bar{R} \Delta\mathbf{u} \tag{36}$$

Combining (34), we can further represent the cost function as follows:

$$\begin{aligned} J(k) &= (S\Delta\hat{x}(k) + T\Delta\mathbf{u})^T\bar{Q}(S\Delta\hat{x}(k) + T\Delta\mathbf{u}) + \Delta\mathbf{u}^T\bar{R}\Delta\mathbf{u} \\ &= \Delta\mathbf{u}^T T^T\bar{Q}T\Delta\mathbf{u} + \Delta\hat{x}(k)^T S^T\bar{Q}S\Delta\hat{x}(k) \\ &+ \Delta\hat{x}(k)^T S^T\bar{Q}T\Delta\mathbf{u} + \Delta\mathbf{u}^T T^T\bar{Q}S\Delta\hat{x}(k) + \Delta\mathbf{u}^T\bar{R}\Delta\mathbf{u} \end{aligned} \tag{37}$$

In (37), the value of S, T, Q, R and $\Delta x(k)$ is known. Our objective is to minimize the cost function. Therefore, we can disregard the constant term in the cost function. The problem can be reformulated as follows:

$$\begin{aligned} \min_{\Delta\mathbf{u}} \ & \Delta\mathbf{u}^T(T^T\bar{Q}T + \bar{R})\Delta\mathbf{u} + 2\Delta\hat{x}(k)^T S^T\bar{Q}T\mathbf{u} \\ &= \Delta\mathbf{u}^T G\Delta\mathbf{u} + H\Delta\mathbf{u} \end{aligned} \tag{38}$$

Additionally, the absolute value of the inputs should not exceed 1. Hence, we have set the following constraint:

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & & 1 \\ -1 & & & & & \\ & -1 & & & & \\ & & -1 & & & \\ & & & \ddots & & \\ & & & & & -1 \end{bmatrix}}_{M} \Delta\mathbf{u} \leq \underbrace{\begin{bmatrix} 1-u_{eq} \\ 1-u_{eq} \\ 1-u_{eq} \\ \vdots \\ 1-u_{eq} \\ 1+u_{eq} \\ 1+u_{eq} \\ 1+u_{eq} \\ \vdots \\ 1+u_{eq} \end{bmatrix}}_{F} \tag{39}$$

Combine (38) and (39), we can get a quadratic programming problem:

$$\begin{aligned} \min_{\Delta\mathbf{u}} \ & \Delta\mathbf{u}^T G\Delta\mathbf{u} + H\Delta\mathbf{u} \\ & s.t.\ M\Delta\mathbf{u} \leq F \end{aligned} \tag{40}$$

In Simulink, we use the function **quadprog** to calculate the optimal input sequence at each step and pick the first input as the control input in next step, which is then applied to the plant.

The block diagram of controlled system is shown in Figure 12.

2) Reference tracking: In output feedback MPC for reference tracking, the cost function is designed to minimize the deviation between the system's true state and a desired reference trajectory.

The objective function is constructed based on tracking error and control effort in the quadratic form as:

$$\begin{aligned} J(k) &= \sum_{i=k}^{i=k+N_p-1} l(\hat{x} - x_{ref}, u - u_{ref}) \\ &= \sum_{i=k}^{i=k+N_p-1} (\hat{x} - x_{ref})^T Q(\hat{x} - x_{ref}) + (u - u_{eq})^T R(u - u_{eq}) \end{aligned} \tag{41}$$

We only have the reference information of α. Although the reference value for α may vary over time, its rate of change is relatively low. If we consider a short horizon, we can approximate the system's

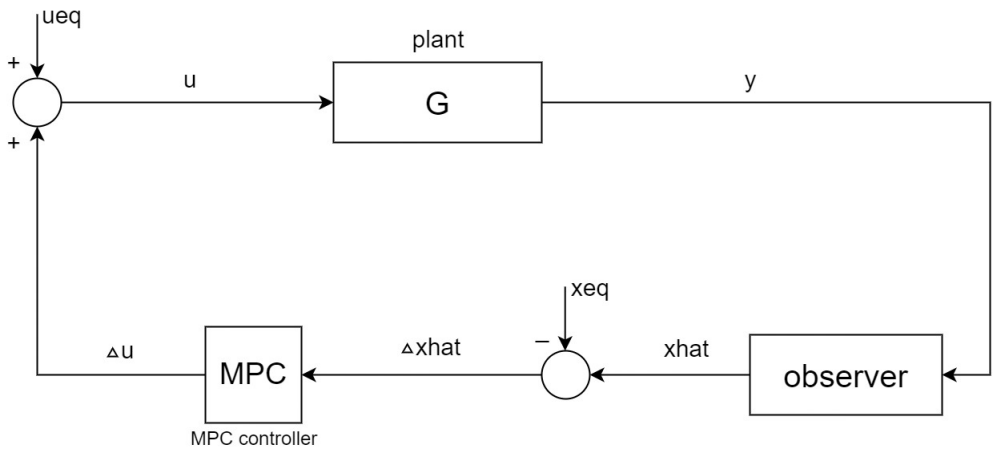


Fig. 12. The structure of MPC controller under the case of stabilizing at one equilibrium.

motion as stabilize around a certain point. Based on system dynamic and reference α, we can calculate the reference values for other states and the input:

$$x_{ref}(2) = \alpha_{ref} \tag{42}$$

$$x_{ref}(1) = \sqrt{|\frac{-p_5}{p_6} sin(\alpha_{ref})|} \tag{43}$$

$$x_{ref}(3) = 0 \tag{44}$$

$$u_{ref} = \frac{1}{-p_3}(p_1 x_{ref}(1) + p_2 x_{ref}(1)^2) \tag{45}$$

After calculating all reference values, the cost function can be represented as:

$$J(k) = (\mathbf{x} - \mathbf{x}_{ref})^T \bar{Q} (\mathbf{x} - \mathbf{x}_{ref}) + (\mathbf{u} - \mathbf{u}_{ref})^T \bar{R} (\mathbf{u} - \mathbf{u}_{ref}) \tag{46}$$

In general, $\mathbf{x}_{ref}$ and $\mathbf{u}_{ref}$ is vectors containing the future N_p reference values for states and inputs. However, since we have chosen a relatively small prediction horizon and the changing rate of reference is low, we can assume that all reference values in the vector are equal to the first reference value. This approach can indeed reduce the computational load of MPC since we only need to consider a single reference value for the entire prediction horizon.

$$\mathbf{x}_{ref} = \begin{bmatrix} x_{ref}(k) \\ x_{ref}(k) \\ \vdots \\ x_{ref}(k) \end{bmatrix} \quad \mathbf{u}_{ref} = \begin{bmatrix} u_{ref}(k) \\ u_{ref}(k) \\ \vdots \\ u_{ref}(k) \end{bmatrix}$$

Similarly with the previous section, we can simplify the problem to a quadratic programming problem:

$$\min_{\mathbf{u}} \quad \mathbf{u}^T(T^T\bar{Q}T+\bar{R})\mathbf{u}+2(\hat{x}(k)^TS^T\bar{Q}T+\mathbf{x}_{ref}^T\bar{Q}T+\mathbf{u}_{ref}^T\bar{R})\mathbf{u}$$

$$s.t. \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ -1 & & & & & \\ & -1 & & & & \\ & & -1 & & & \\ & & & \ddots & & \\ & & & & & -1 \end{bmatrix} \mathbf{u} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{47}$$

Then we utilize the **quadprog** function to compute the optimal input sequence and extract the first input as the control input.

The block diagram of controlled system is shown in Figure 13.

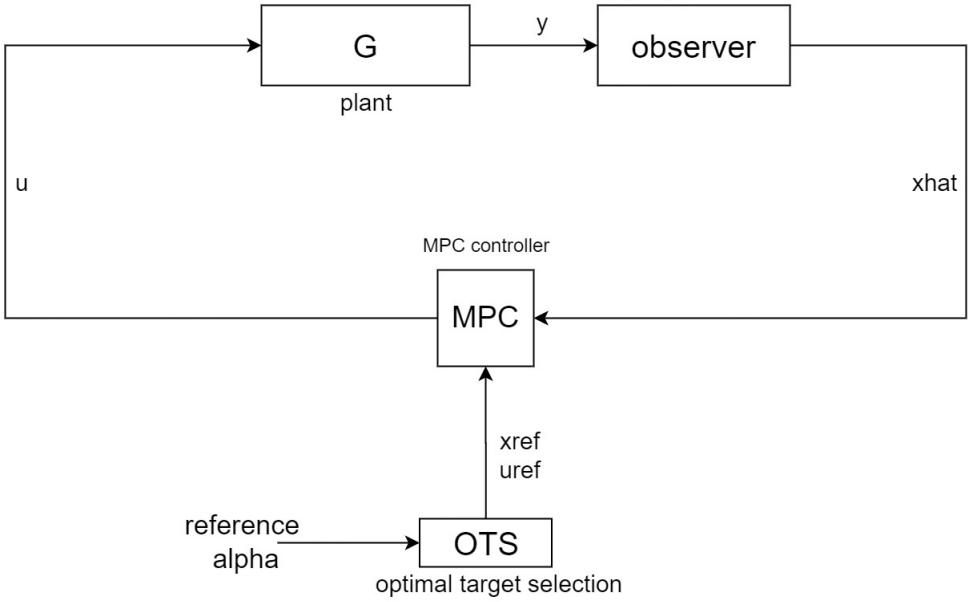


Fig. 13. The structure of MPC controller under the case of reference tracking.

3) Tune Q, R and N_p: Just like the control objectives in designing an LQR controller, the control objectives of an MPC controller are also to achieve system stability, eliminate offset, minimize overshoot and settling time, and ensure the system can return to the steady state in the presence of disturbances. In reference tracking case, the system needs to be able to track the reference value accurately while maintaining stable and smooth operation.

Firstly, we tune the length of prediction horizon N_p and observe the system performance. When the prediction horizon is short, the controller can only predict the system's behavior in the near future. As a result, it cannot fully consider the long-term dynamic characteristics and trends of the system, which limits its control performance. The settling time of the system tends to be larger, and the overshoot is also relatively higher in this case.

When the prediction horizon is increased, the settling time and overshoot of the system both decrease, leading to an improvement in control performance. However, when the prediction horizon becomes too large, the system can exhibit oscillations in the presence of disturbances after reaching stability. This is because a larger prediction horizon makes the controller more sensitive to disturbances, and even small disturbances can be amplified in the future predictions, affecting the controller's performance. Additionally, if the horizon is too large, errors in the system model and measurement noises can accumulate over the horizon, leading to a decrease in the accuracy of long-term predictions. Finally, we set $N_p = 10$.

The parameters in Q and R weighting matrices have the same impact on the control performance, as well as the process of adjusting them, as in the case of LQR controller. We first test the control performance under different values of $\frac{q_2}{r}$ to find an appropriate ratio value. After that, we adjusted value of q_1 and q_3 to further improve the system performance. In the end, we selected the following parameters:

$$Q = diag[5 \quad 120 \quad 3], \; R = 10, \; N_p = 10 \tag{48}$$

However, this set of parameters didn't work well in case of reference tracking. The actual alpha of the system deviates from the reference value sometimes, as shown in Figure 14.

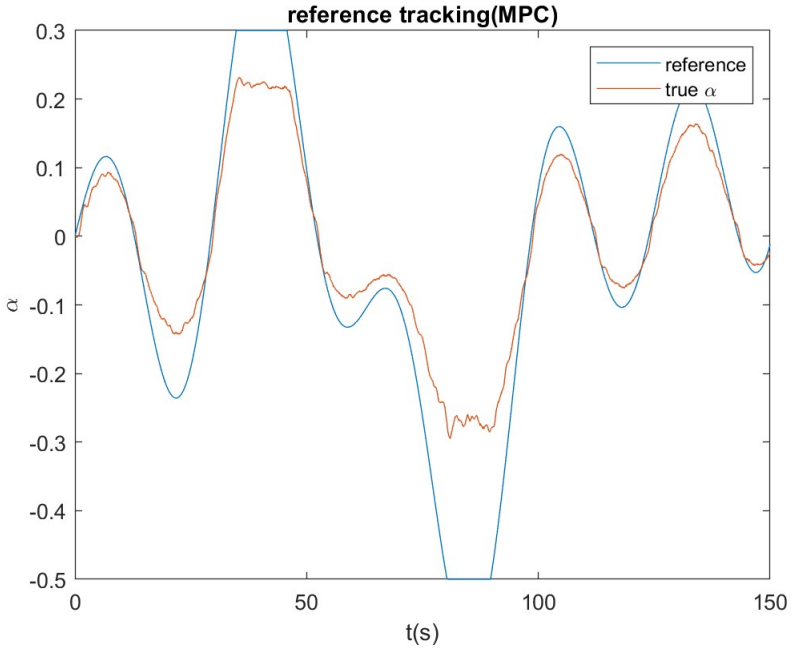


Fig. 14. reference tracking

This could be due to the assumption made in MPC where we assume that the future references in the reference sequence are all equal to the first reference in the sequence. As a result, if the prediction horizon is large, it can lead to a decrease in control performance, especially when there are significant changes in the reference values in a short period. Therefore, we attempted to decrease the prediction horizon. The results showed that using a smaller prediction horizon improved the performance of the controller while reference tracking.

The parameters for reference tracking are shown below:

$$Q = diag[5 \quad 120 \quad 3], \; R = 10, \; N_p = 3 \tag{49}$$

VII. Results evaluation and discussion

A. Simulation results

1) stabilizing at one equilibrium(LQR): We set the stabilizing point at $\alpha = -0.5$. The simulation result is shown in Figure 15. From the graph, it can be observed that the system settling time is 1.45 seconds, and the overshoot is 0.09.

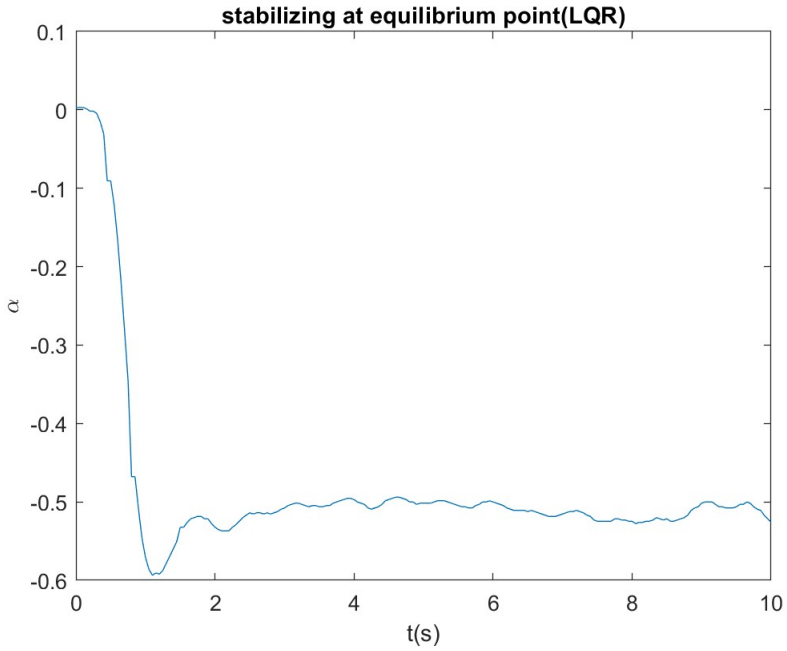


Fig. 15. stabilizing at one equilibrium(LQR)

2) stabilizing at one equilibrium(MPC): Same as last experiment, the stabilizing point is at $\alpha = -0.5$. The simulation result is shown in Figure 16. From the figure, it can be observed that the settling time is 2.8 seconds, and the overshoot is 0.34.

3) reference tracking(LQR): We have designed a α reference trajectory that includes a wide range of α values, including both rapidly changing sections and stable sections at specific points. This trajectory aims to comprehensively test the controller's ability to track the reference.The simulation result is shown in Figure 17.

4) reference tracking(MPC): The simulation result is shown in Figure 18.

B. Discussion

By comparing Figure 15 and Figure 16, we can observe that in the stabilizing case, the LQR controller exhibits smaller overshoot and settling time compared to the MPC controller. However, after the system has stabilized, if a disturbance occurs, the LQR-controlled system experiences larger fluctuations, whereas the MPC-controlled system demonstrates better stability and robustness to disturbance.

The reason why LQR control exhibits smaller overshoot and settling time is because LQR mathematical theory is comprehensive, and the computed control input is the optimal input under the current conditions. Therefore, it allows the system to quickly reach the steady state with minimal control action. LQR control is primarily used for linear control. In the case of controlling a nonlinear system to its equilibrium point, we approximate the nonlinear system as a linear system in the vicinity of this equilibrium point. Therefore the LQR controller can apply to this case.

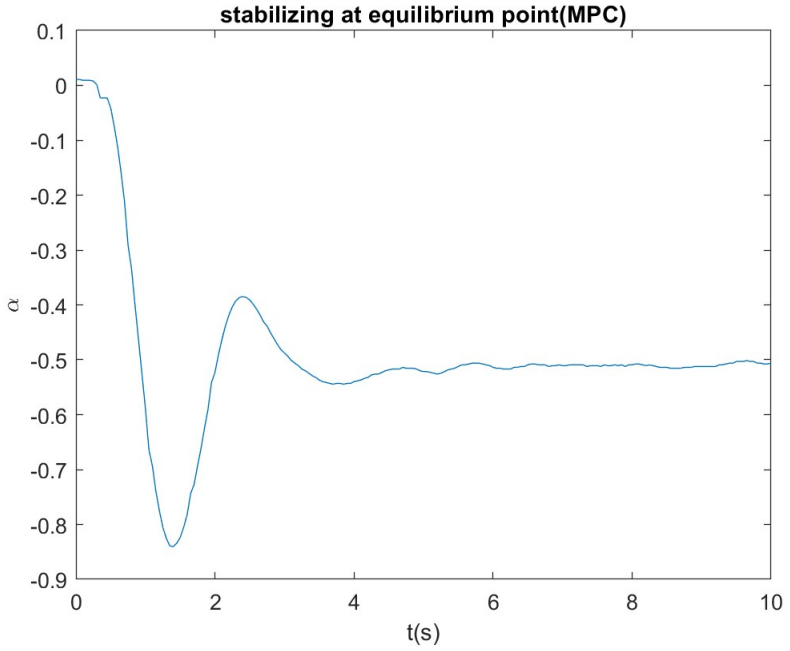


Fig. 16. stabilizing at one equilibrium(MPC)

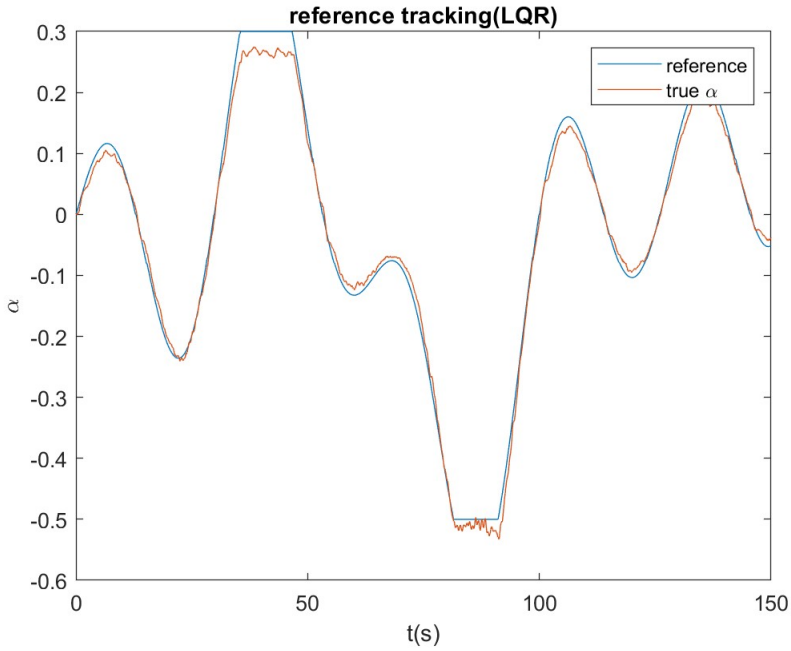


Fig. 17. reference tracking(LQR)

However, after the system reaches a stable state, LQR control may exhibit poorer disturbance rejection capability. If a disturbance occurs, the system may experience small oscillations. This is because LQR control is based on the assumption of system linearization. If the system deviates too far from the equilibrium point, there may be errors between the linearized system and the original nonlinear system, leading to a degradation in controller performance. Also, the principle of LQR control is to minimize

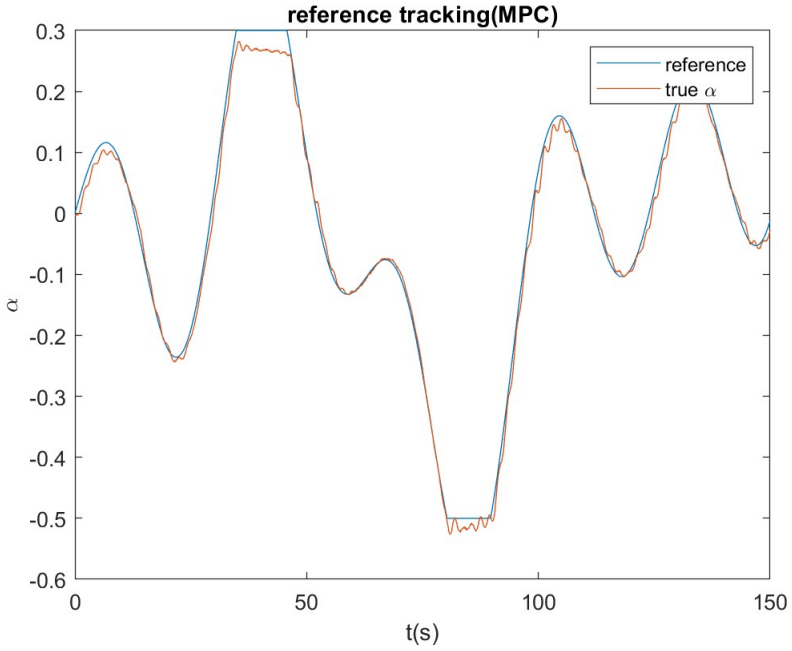


Fig. 18. reference tracking(MPC)

a quadratic function of the states and control inputs, and this performance metric is highly sensitive to disturbances and their magnitude can be amplified by the quadratic function. Since LQR control aims to quickly reduce the value of this quadratic function, it may lead to oscillations in the system. Another reason for the limited disturbance rejection capability of LQR control is that it does not consider future changes in the system's states. LQR control calculates the control input based solely on the current system state. When a disturbance is present, LQR control aims to return to the steady state as quickly as possible based on the current state, without considering whether the system can continue to remain stable at that point. This can result in excessive control action and small oscillations in the system.

Additionally, LQR control requires an accurate model of the system, since LQR control relies on the precise knowledge of the system dynamics for optimal control design. If there are model errors at this equilibrium point, it can lead to undesired offsets and reduced control effectiveness.

As for MPC control, the principle of MPC involves computing the optimal control inputs over a certain prediction horizon. The control input obtained from MPC is not necessarily optimal for the current conditions. As a result, compared to LQR control, MPC control may require more time to reach the steady state, and it may exhibit larger overshoot.

However, MPC exhibits greater robustness after reaching a steady state. In other words, it possesses stronger resistance to disturbances compared to LQR control. This is because MPC control makes control decisions by predicting the future behavior of the system. In the presence of disturbances, MPC not only considers how to bring the system back to the steady state as quickly as possible but also takes into account whether the system can continue to remain stable at this equilibrium point in the future. As a result, MPC is better able to adapt to and suppress disturbances when compared to LQR control.

By comparing Figure 17 and Figure 18, we can observe the performance of LQR and MPC control in reference tracking. Both controllers show similar performance, but overall, the MPC controller performs better in reference tracking. There are several reasons why the MPC controller is more suitable for reference tracking:

1) Firstly, LQR control is primarily designed for linear systems, while MPC control can control nonlinear systems. Since our system is nonlinear, employing LQR control could result in a decrease in control performance. To address this issue, we have employed an interpolation method for the controller gain in LQR control, which estimates a control strategy suitable for nonlinear systems. This is also the reason why there is small difference in the performance between the two control methods.
2) Furthermore, MPC achieves better control of dynamic systems by predicting the system's future changes over a certain period of time and executing control strategies accordingly. For instance, within the 50-70 seconds range shown in the figure, the reference value undergoes significant short-term variations. In such cases, MPC demonstrates better control performance, enabling the system's actual trajectory to closely align with the reference trajectory.
3) Additionally, for MPC control, it is relatively easy to incorporate constraints and consider them in the optimization problem. For example, in this assignment, we have a constraint that limits the absolute value of the input to be within 1. On the other hand, adding constraints to LQR control is challenging. We can only apply a **saturation** module to restrict the control input after obtaining it, which leads to suboptimal inputs to the actual plant. In the case of reference tracking, where our α reference may undergo significant variations, the control inputs obtained from LQR control are likely to exceed the constraint range. As a result, the control performance of LQR is inferior to that of MPC.

It is worth noting that both LQR and MPC control exhibit some deviation when the reference is at 0.3. This could be due to inaccuracies in the model at that specific point, causing both control methods to fail in accurately tracking the reference.

In conclusion, LQR control is more suitable for situations where a quick control response to reach a specific point is required. On the other hand, MPC control is better suited for complex control scenarios, such as systems with various constraints or tracking complex reference. It is also suitable for situations where the system needs to maintain stability at a particular point over a long period of time.

REFERENCES

[1] Propeller theory. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/w/index.php?title=Propeller_theory&oldid=1153805641

[2] idnlgrey, Nonlinear grey-box mode. MathWorks. (2023, June).Retrieved from https://nl.mathworks.com/help/ident/ref/idnlgrey.html

[3] Pole placement design. MathWorks. (2023, June).Retrieved from https://nl.mathworks.com/help/control/ref/place.html

[4] PID controller - MATLAB & Simulink. (2023, May 15). Retrieved from https://nl.mathworks.com/help/simulink/slref/pidcontroller.html