Assignment for EE5101/ME5401 Linear Systems

AY2024/2025 Semester 2

Control of a Stationary Self-Balancing Two-wheeled Vehicle

XU NUO

A0313771H

e1499166@u.nus.edu

April 24, 2025

Abstract. This project investigates multivariable control of a two-input, three-output system using state-space techniques. An LQR-based controller and a full-order observer are designed to achieve stable output tracking and decoupling. Integral action is introduced via system augmentation to ensure zero steady-state error in the presence of constant disturbances. Simulations validate the controller's ability to track a given operating set point while rejecting disturbances. However, theoretical analysis confirms that arbitrary set point tracking with zero steady-state error is not achievable due to the system's underactuated nature.

Key words: Pole Placement, LQR, Observer, Decoupling Control, Integral Control

Content

1 Introduction

The stationary self-balancing two-wheeled vehicle is an advanced system inspired by the challenges of riding a bicycle. It aims to achieve autonomous balance without human input, particularly while remaining stationary. In this project, only the self-balancing behavior when the vehicle is stationary is considered.

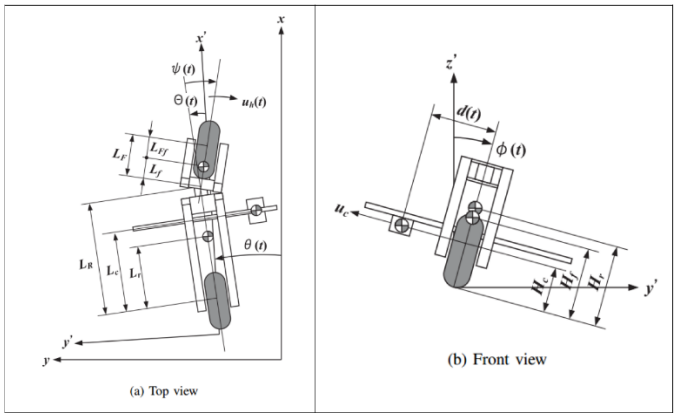


Fig. 1. Two-wheeled Vehicle Structure Model

The mechanical structure for the two-wheeled vehicle is given in Figure 1. The two-wheeled vehicle is stabilized by moving the cart position $d(t)$ and adjusting the handle angle $\psi(t)$. The control inputs are the voltages $u_c(t)$ and $u_h(t)$ to two DC servo motors, which drives the cart system and the steering system correspondingly.

The state space linear model for the two-wheeled vehicle is derived to be

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

where the state variable is

$$x = \begin{bmatrix} d(t) & \phi(t) & \psi(t) & \dot{d}(t) & \dot{\phi}(t) & \dot{\psi}(t) \end{bmatrix}^T \tag{2}$$

My matriculation number is A0313771H. The unknown variables contained in the parameters are $a = 3,\ b = 7,\ c = 7$ and $d = 1$. After calculation, the matrices and the input vector are shown below.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 6.5 & -10 & -18 & 0 & 0 \\ -21.5185 & 18.0252 & 5.0828 & -3.8733 & -4.2022 & 0.3149 \\ 5 & -3.6 & 0 & 0 & 0 & -11.1364 \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 27 & 11.2 \\ 5.81 & -1.7364 \\ 40 & 61.4 \end{bmatrix} \tag{4}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad u = [u_c(t), u_h(t)]^T \tag{5}$$

The initial condition for the two-wheeled vehicle system is assumed to be

$$x_0 = [0.2, -0.1, 0.15, -1, 0.8, 0]^T \tag{6}$$

Each of the following sections corresponds to a specific task in this project, and they will be explained one by one. Each section consists of two main components: The first part typically presents the system design and methodology, integrating data and formulas for explanation. The second part generally includes the output obtained from running MATLAB code and the simulation results from the Simulink model, accompanied by an analysis of the results.

2 Pole Placement State Feedback Controller Design

2.1 Design Methodology

Every system has its design specifications, this two-wheeled vehicle is no exception. The transient response performance specifications for all the outputs y in state space model are as follows:

1) The overshoot is less than 10%
2) The 2% settling time is less than 5 seconds.

A standard second-order transfer function is derived as

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{7}$$

where ω_n is undamped natural frequency, and ζ is damping ratio.

Taken the design specifications into consideration, the maximum overshoot M_p and settling time t_s can be calculated

$$M_p = e^{-\frac{\pi\zeta}{\sqrt{(1-\zeta^2)}}} < 10\% \tag{8}$$

$$t_s \cong \frac{4.0}{\zeta\omega_n} < 5 \tag{9}$$

By substituting $\zeta = 0.8$ and $\omega_n = 1.5$ into the formulas for maximum overshoot and settling time, it can be verified that the standard second-order system meets the design specifications required by the task. So, the dominant poles of this second-order system can be expressed as

$$p_1, p_2 = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -1.2 \pm 0.9j \tag{10}$$

As mentioned in the slides in Chapter 7, the extra poles other than the dominant ones should be located 2-5 times faster than the dominant ones. So, I place the poles as shown in Table 2-1

p_1	p_2	p_3	p_4	p_5	p_6
$-1.2 + 0.9j$	$-1.2 - 0.9j$	-4	-4	-6	-6

TABLE 2-1. Pole Placement

In state-space control theory, we can arbitrarily assign the closed-loop poles using state feedback $u = -Kx$ only when the system is controllable. To ensure that the designed state feedback controller is meaningful and realizable, we must first verify controllability. The controllability matrix is constructed as follows:

$$\begin{aligned} W_c &= [B, AB, A^2B, A^3B, A^4B, A^5B] \\ &= [b_1, b_2, Ab_1, Ab_2, A^2b_1, A^2b_2, \dots, A^5b_1, A^5b_2] \end{aligned} \tag{11}$$

The rank of W_c is 6, which means the system is controllable. As mentioned in slides of Chapter 7, for MIMO system, we need to select the n independent vectors out of nm vectors from the controllability matrix in the strict order from left to right, and group them in a square matrix C in the following form

$$C_{matrix} = [b_1, Ab_1, A^2b_1, b_2, Ab_2, A^2b_2] \tag{12}$$

Once we have the matrix C, we need to compute the inverse of C

$$C_{matrix}^{-1} = \begin{bmatrix} q_1^T \\ q_2^T \\ q_3^T \\ q_4^T \\ q_5^T \\ q_6^T \end{bmatrix} \tag{13}$$

The system has a state dimension of $n = 6$. And it has a control input $u = [u_c, u_h]^T$, which means that it is a 2-input system and has two control channels. I assign three poles to each control channel, so the system is divided into two subsystems of dimension three. The first subsystem has $d_1 = 3$ states, controlled by u_c. The second subsystem has $d_2 = 3$ states, controlled by u_h. Therefore, the transformation matrix T is constructed as shown below.

$$T = \begin{bmatrix} q_3^T \\ q_3^T A \\ q_3^T A^2 \\ q_6^T \\ q_6^T A \\ q_6^T A^2 \end{bmatrix} \tag{14}$$

Then the system can be transferred into a controllable canonical form

$$\bar{A} = TAT^{-1}, \quad \bar{B} = TB \tag{15}$$

There are two inputs, so there are two non-trivial rows since the inputs only affect the non-trivial rows. The feedback gain matrix for the controllable canonical form is

$$\bar{K} = \begin{bmatrix} \bar{k}_{11} & \bar{k}_{12} & \bar{k}_{13} & \bar{k}_{14} & \bar{k}_{15} & \bar{k}_{16} \\ \bar{k}_{21} & \bar{k}_{22} & \bar{k}_{23} & \bar{k}_{24} & \bar{k}_{25} & \bar{k}_{26} \end{bmatrix} \tag{16}$$

A_d is the closed-loop state matrix of the system in controllable canonical form coordinates. For A_d, we aim for it to take the following structure.

$$A_d = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -a_{13} & -a_{12} & -a_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -a_{23} & -a_{22} & -a_{21} \end{bmatrix} \tag{17}$$

Here a_{11}, a_{12}, a_{13} and a_{21}, a_{22}, a_{23} represent the coefficients of the characteristic polynomials of two separate third-order subsystems, after placing their poles at p_1, p_2, p_3 and p_4, p_5, p_6 respectively. Specifically, they can be expressed as

$$\begin{aligned} (s-p_1)(s-p_2)(s-p_3) &= s^3 + a_{11}s^2 + a_{12}s + a_{13} \\ (s-p_4)(s-p_5)(s-p_6) &= s^3 + a_{21}s^2 + a_{22}s + a_{23} \end{aligned} \tag{18}$$

The closed-loop state matrix can also be expressed as

$$A_d = \bar{A} - \bar{B}\bar{K} \tag{19}$$

Solving the equations symbolically yields the gain matrix $\bar{K}$ in the controllable canonical form by comparing A_d and $\bar{A} - \bar{B}\bar{K}$, which can then be transformed back to the original coordinate system to obtain the feedback gain K using the transformation matrix T as follows.

$$K = \bar{K}T \tag{20}$$

$$K = \begin{bmatrix} -7.7640 & 6.8301 & 2.5855 & -3.4838 & 2.1577 & 1.1726 \\ -30.5729 & 30.1138 & 12.2162 & -2.5960 & 4.6458 & 0.9907 \end{bmatrix} \tag{21}$$

2.2 Results and Analysis

The real-time variations of the output, state, and control signals under the system's free response—starting from the initial state x_0 with no external input—are shown in the figures below.

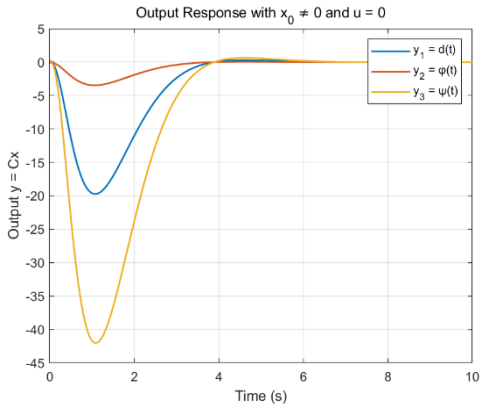


Fig. 2-1. Output of Free Response

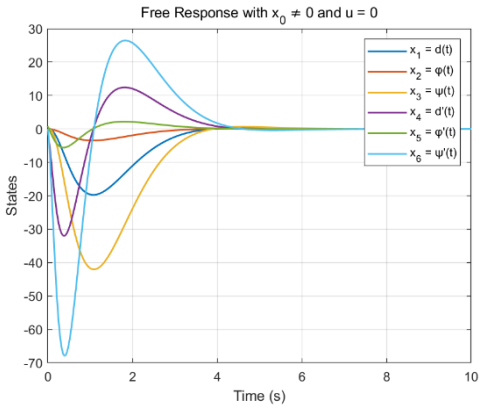


Fig. 2-2. States of Free Response

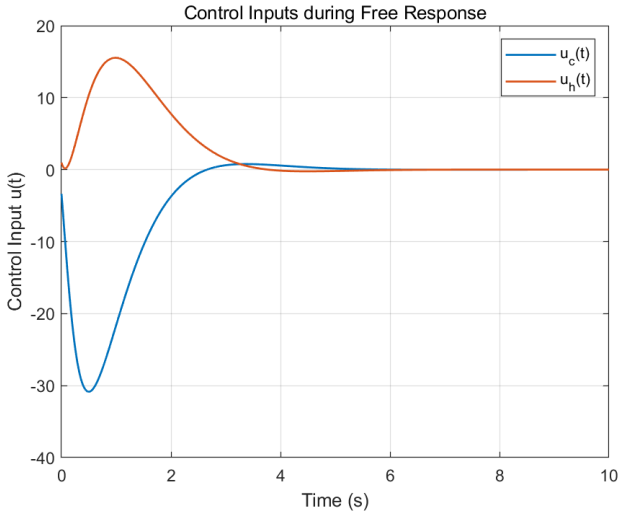


Fig. 2-3. Control of Free Response

The real-time output responses under step inputs $r = [0,1]$ and $[1,0]$, with an initial state of 0 are also illustrated in the figures below.

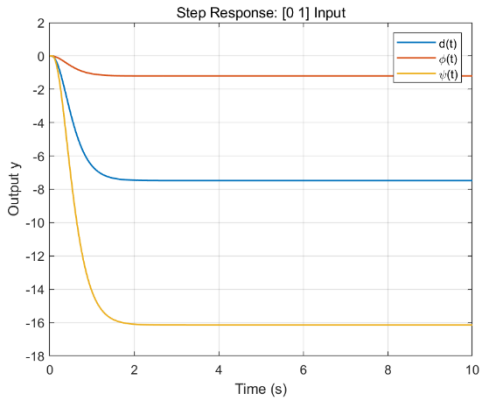


Fig. 2-4. Output of Step Response [0,1]

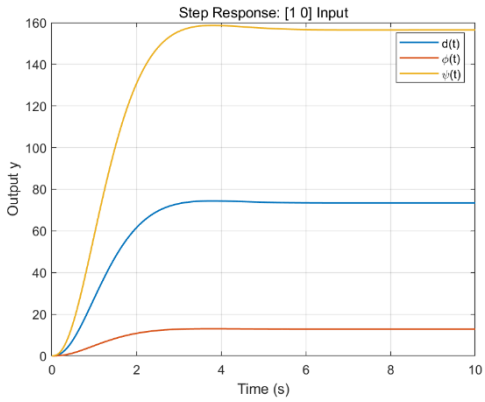


Fig. 2-5. Output of Step Response [1,0]

The program output obtained by running the MATLAB script *task1.m* is shown below.

```
Max control input during free response:
u_c: 30.85
u_h: 15.54

=== Performance Report for Input [1, 0] ===
Output: d(t)   | Overshoot:    1.37% | Settling Time:  2.79s
Output: phi(t) | Overshoot:    1.37% | Settling Time:  2.78s
Output: psi(t) | Overshoot:    1.37% | Settling Time:  2.80s

=== Performance Report for Input [0, 1] ===
Output: d(t)   | Overshoot:    0.00% | Settling Time:  1.50s
Output: phi(t) | Overshoot:    0.00% | Settling Time:  1.48s
Output: psi(t) | Overshoot:    0.00% | Settling Time:  1.52s
```

Based on the output above and Figure 2-4 and 2-5, it can be concluded that the pole placement fully meets the transient response requirements—namely, the overshoot is less than 10% and the settling time is less than 5 seconds. The max control inputs during free response are $u_c = 30.85$ and $u_h = 15.54$ respectively. The size of the control signals is reasonable.

I also built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task1.m*. The block diagrams corresponding to the free response and step response are shown below.

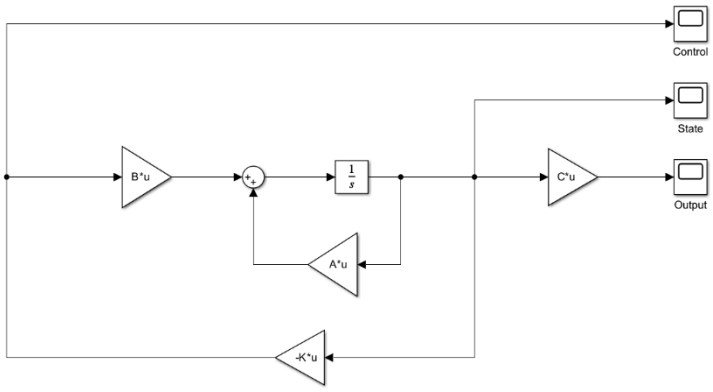


Fig. 2-6. Model of Free Response Using Pole Placement Method

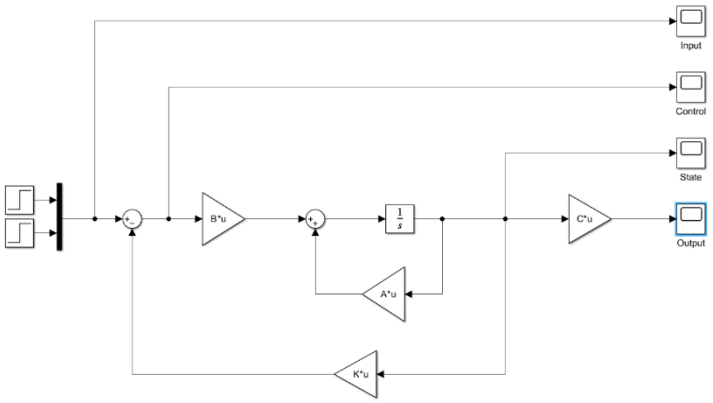


Fig. 2-7. Model of Step Response Using Pole Placement Method

The corresponding observation results by the scopes are shown in the following figures.

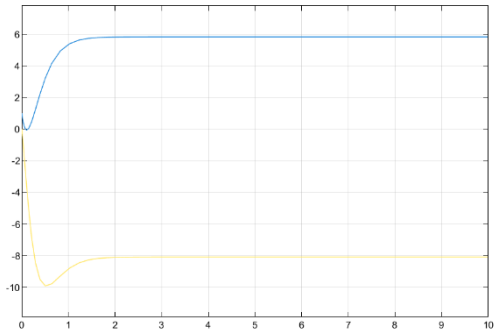


Fig. 2-8. Control of Step Response [0,1] (SIM)

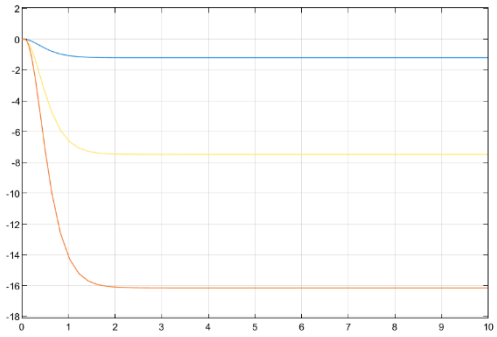


Fig. 2-9. Output of Step Response [0,1] (SIM)

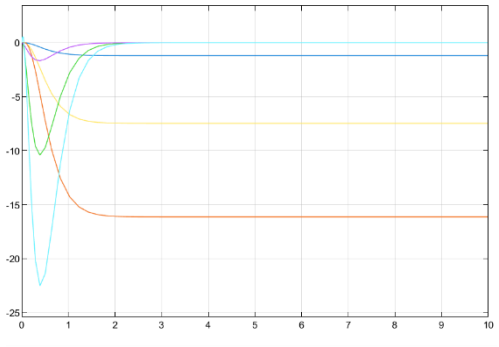


Fig. 2-10. States of Step Response [0,1] (SIM)

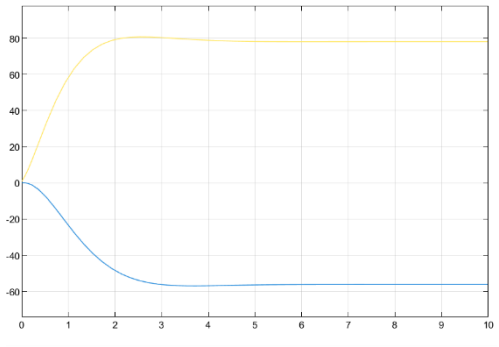


Fig. 2-11. Control of Step Response [1,0] (SIM)

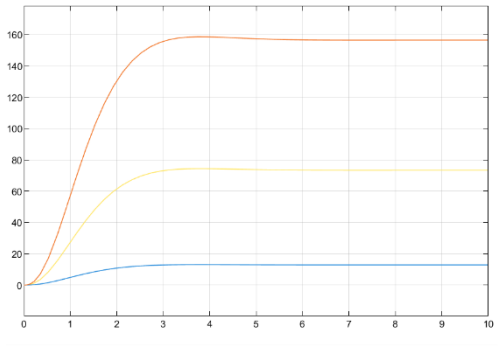


Fig. 2-12. Output of Step Response [1,0] (SIM)

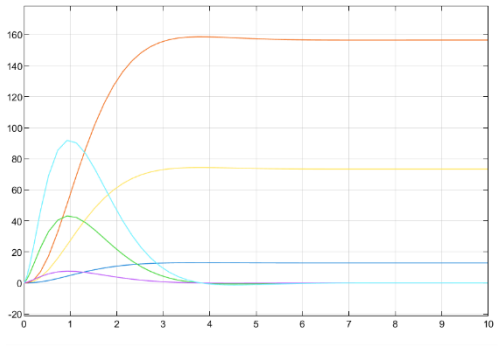


Fig. 2-13. States of Step Response [1,0] (SIM)

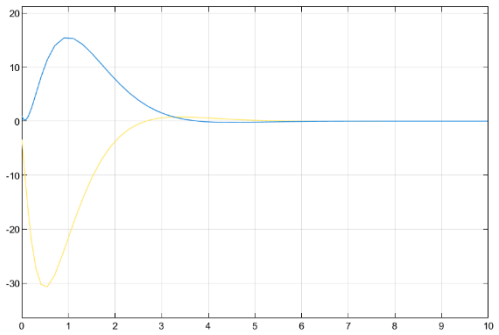


Fig. 2-14. Control of Step Response [0,1] (SIM)

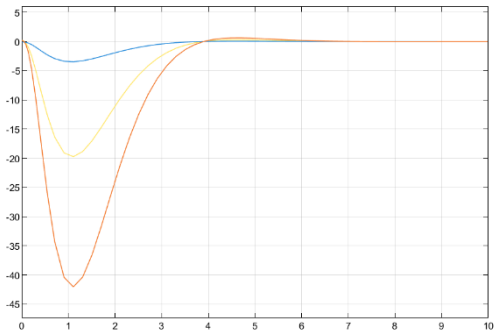


Fig. 2-15. Output of Free Response (SIM)

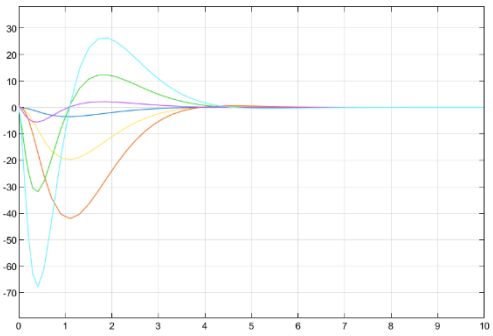


Fig. 2-16. States of Free Response (SIM)

As can be seen by comparing with the figures above, the results obtained from running the MATLAB script *task1.m* are fully consistent with those obtained from the scopes in the SIMULINK simulation. This confirms that both the implementation of the code and the structure of the SIMULINK model diagram are correct.

To discuss the effects of the positions of the poles on system performance, I set up different pole configurations. Since the transient response of the system is primarily determined by the dominant poles, I varied the pair of complex poles. The different pole combinations are listed in the table below.

p_1	p_2	p_3	p_4	p_5	p_6
$-0.5+0.9j$	$-0.5-0.9j$	-4	-4	-6	-6
$-2+0.9j$	$-2-0.9j$	-4	-4	-6	-6
$-1.2+0.2j$	$-1.2-0.2j$	-4	-4	-6	-6
$-1.2+2j$	$-1.2-2j$	-4	-4	-6	-6

TABLE 2-2. Different Pole Configurations

The real-time variations of the output, state, and control signals under the system's free response—starting from the initial state x_0 with no external input. The real-time output responses under step inputs $r=[0,1]$ and $[1,0]$, with an initial state of 0, for different pole configurations are all shown in the figures below.

Configuration 1: $p_{1,2}=-0.5\pm0.9j$

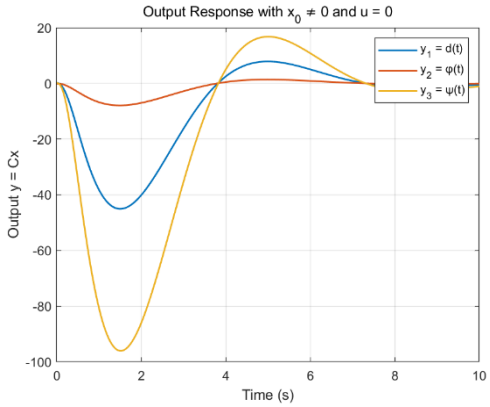


Fig. 2-17. Output of Free Response (Config. 1)

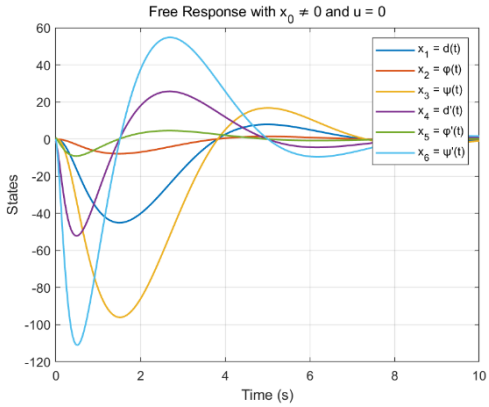


Fig. 2-18. States of Free Response (Config. 1)

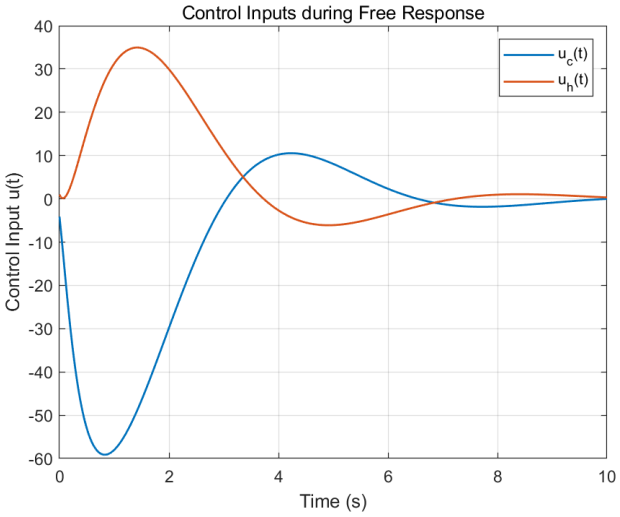
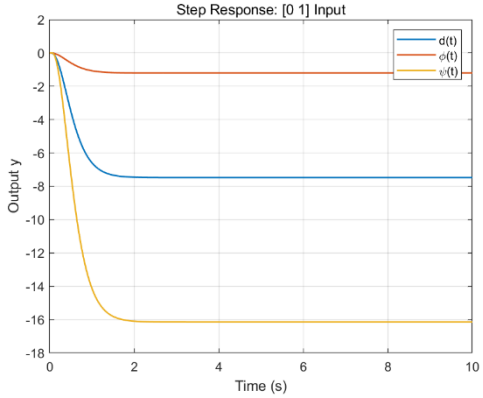


Fig. 2-19. Control of Free Response (Config. 1)



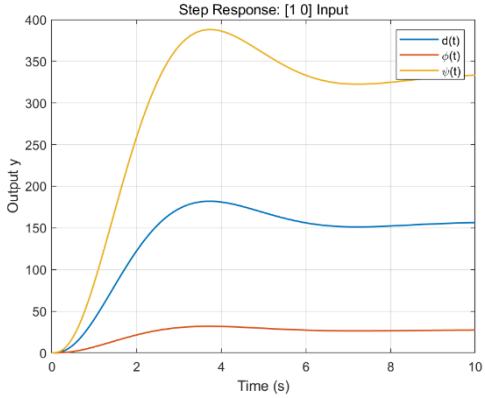
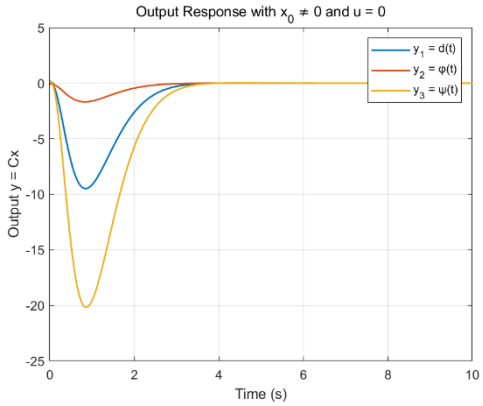


Fig. 2-20/21. Output of Step Response [0,1] and [1,0] (Config. 1)

Configuration 2: $p_{1,2} = -2 \pm 0.9j$



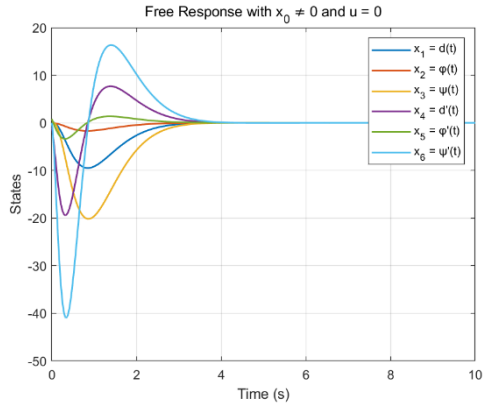


Fig. 2-22. Output of Free Response (Config. 2)

Fig. 2-23. States of Free Response (Config. 2)

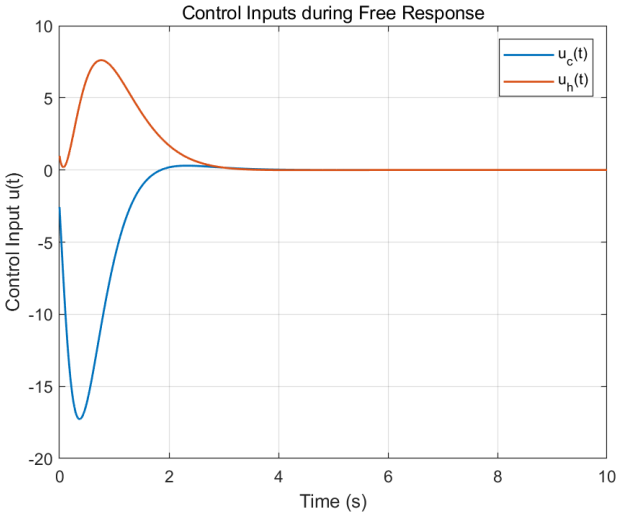


Fig. 2-24. Control of Free Response (Config. 2)

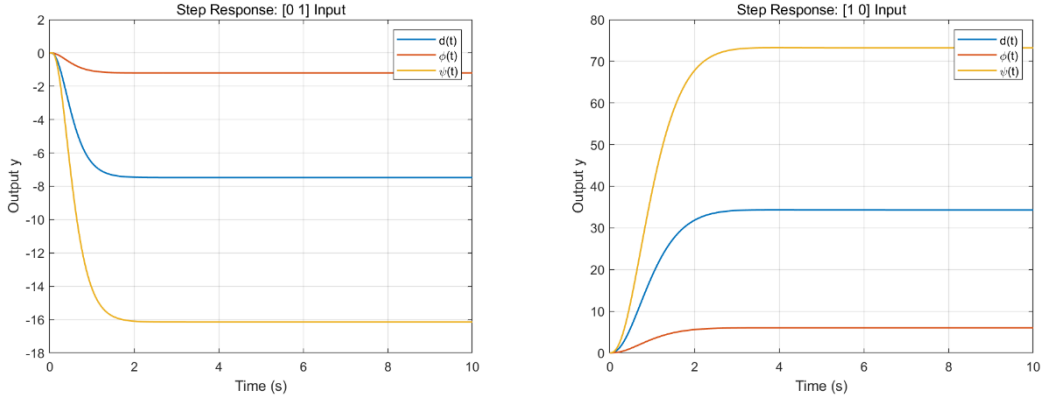


Fig. 2-25/26. Output of Step Response [0,1] and [1,0] (Config. 2)

Configuration 3: $p_{1,2} = -1.2 \pm 0.2j$

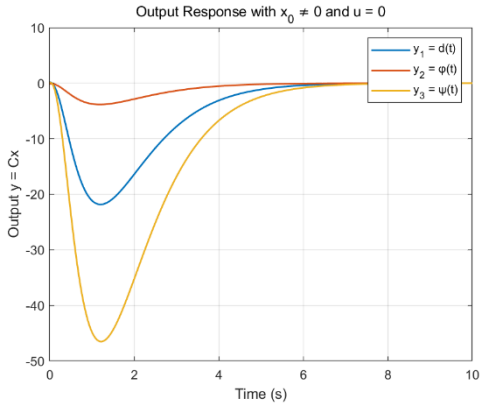


Fig. 2-27. Output of Free Response (Config. 3)

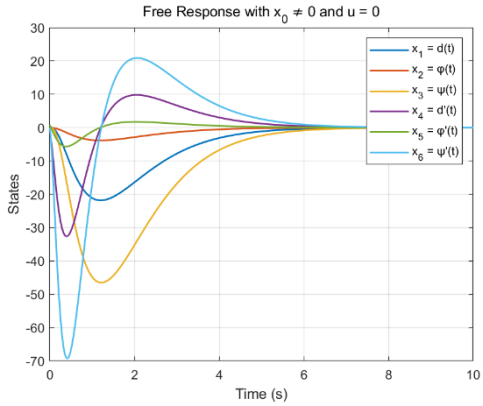


Fig. 2-28. States of Free Response (Config. 3)

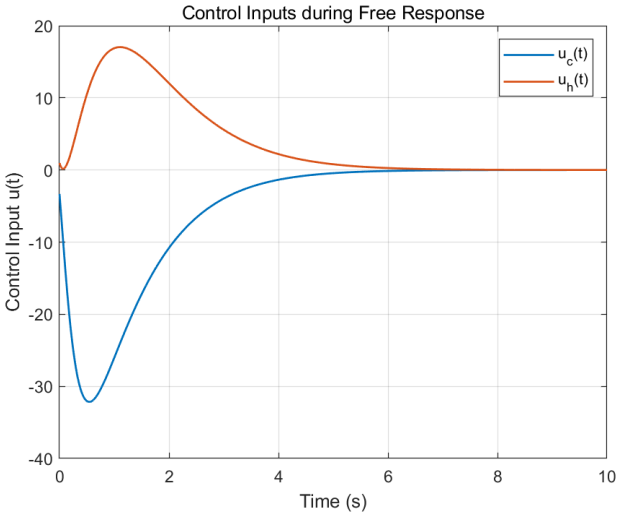
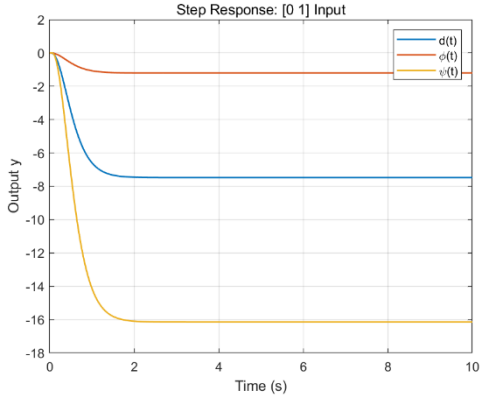


Fig. 2-29. Control of Free Response (Config. 3)



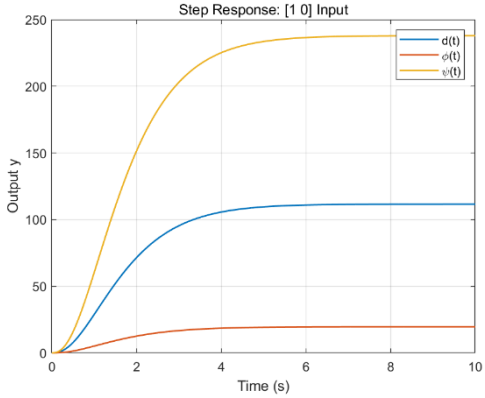


Fig. 2-30/31. Output of Step Response [0,1] and [1,0] (Config. 3)

Configuration 4: $p_{1,2} = -1.2 \pm 2j$

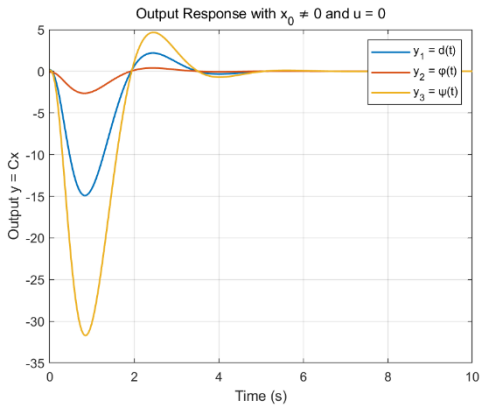


Fig. 2-32. Output of Free Response (Config. 4)

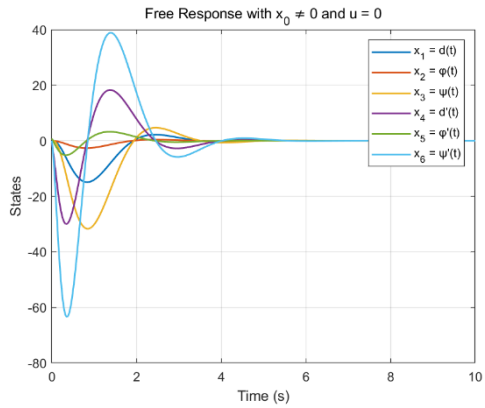


Fig. 2-33. States of Free Response (Config. 4)

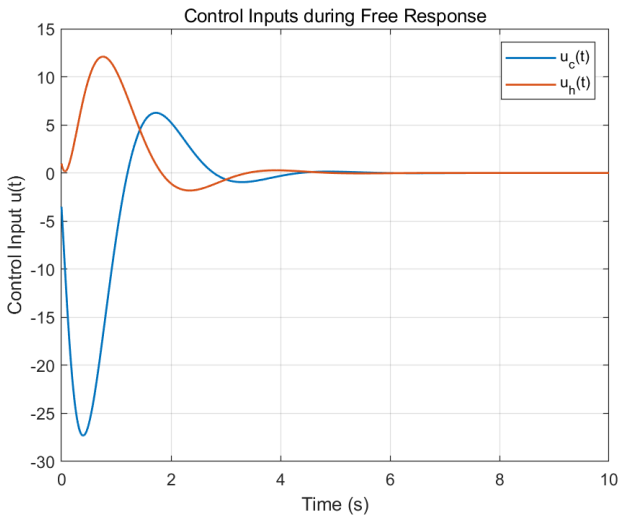


Fig. 2-34. Control of Free Response (Config. 4)

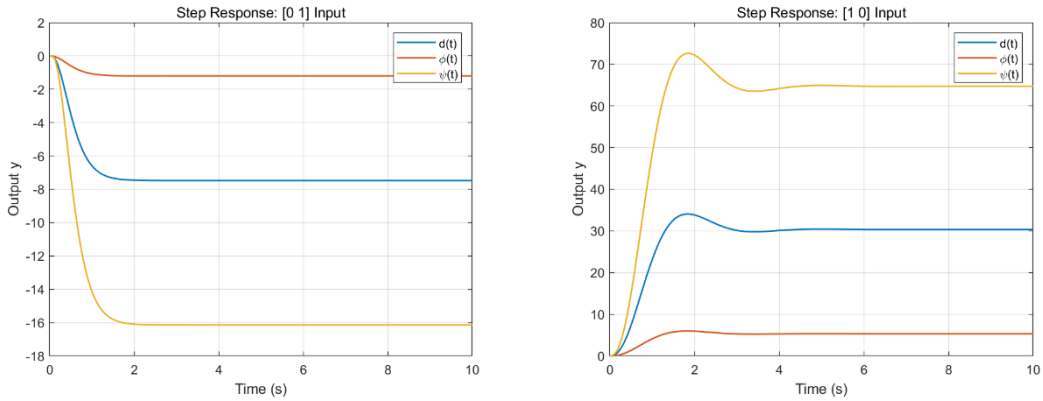


Fig. 2-35/36. Output of Step Response [0,1] and [1,0] (Config. 4)

By adopting a controlled variable approach, I independently increased and decreased the real and imaginary parts of the dominant poles and obtained the corresponding sets of response figures. By comparing the system responses across different configurations, it can be observed that when the real part of the dominant poles becomes more negative, the system responds faster, the settling time becomes shorter, and the oscillation amplitude of some signals tends to be weaker. Conversely, if the real part becomes less negative, the opposite effects occur. Moreover, when the imaginary part of the dominant poles increases, the oscillation frequency of certain signals becomes higher, and the overshoot also increases accordingly; the opposite happens when the imaginary part is reduced.

3 LQR Method State Feedback Controller Design

3.1 Design Methodology

LQR (Linear Quadratic Regulator) is a classic optimal control strategy. Its objective is

to design a feedback law $u(t) = -Kx(t)$, such that the following performance index is minimized

$$J = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u)\, dt \tag{22}$$

where $Q \in \mathbb{R}^{n\times n}$ is the state weighting matrix (positive semi-definite), $R \in \mathbb{R}^{m\times m}$ is the input weighting matrix (positive definite). After repeated trial and error, I set matrix Q and R as follows.

$$Q = \begin{bmatrix} 30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{23}$$

$$R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{24}$$

To obtain the optimal gain matrix K, we solve the Algebraic Riccati Equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{25}$$

Then the optimal gain is given by

$$K = -R^{-1}B^T P \tag{26}$$

where P is the unique positive semi-definite solution to the ARE. To solve the positive definite solution of the ARE, we follow the systematic way of eigenvalue-eigenvector based algorithm. We first define a $2n \times 2n = 12 \times 12$ Hamiltonian matrix

$$\Gamma = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{27}$$

then the eigenvectors and eigenvalues are computed. We select $n = 6$ eigenvectors corresponding to the eigenvalues with negative real parts. This gives us a stable invariant subspace of Γ. Let the selected eigenvectors be stacked into a new matrix as below.

$$\begin{bmatrix} V \\ \mu \end{bmatrix} \tag{28}$$

where $V \in \mathbb{R}^{n\times n}$ (top half) and $\mu \in \mathbb{R}^{n\times n}$ (bottom half). Then the solution to the ARE is given by

$$P = \mu V^{-1} \tag{29}$$

$$P = \begin{bmatrix} 398.9159 & -358.9867 & -160.4762 & 27.5790 & -60.0630 & -8.1878 \\ -358.9687 & 360.2764 & 149.6883 & -25.7517 & 58.3920 & 7.6125 \\ -160.4762 & 149.6883 & 75.7163 & -11.4252 & 24.9144 & 3.4892 \\ 27.5790 & -25.7517 & -11.4252 & 2.1645 & -4.3870 & -0.6274 \\ -60.0630 & 58.3920 & 24.9144 & -4.3870 & 10.2267 & 1.2870 \\ -8.1878 & 7.6125 & 3.4892 & -0.6274 & 1.2870 & 0.2731 \end{bmatrix} \quad (30)$$

Once we get P, the optimal gain matrix K can be calculated.

$$K = \begin{bmatrix} 13.6308 & -10.3080 & -4.8324 & 1.5716 & -1.5102 & 0.2928 \\ -17.9112 & 15.5190 & 8.6025 & -1.3320 & 2.4261 & 1.5018 \end{bmatrix} \quad (31)$$

3.2 Results and Analysis

The real-time variations of the output, state, and control signals under the system's free response—starting from the initial state x_0 with no external input—are shown in the figures below.

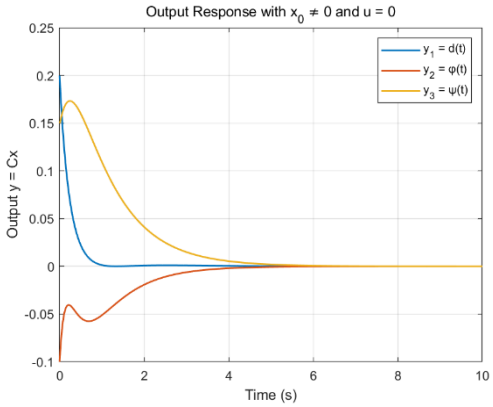


Fig. 3-1. Output of Free Response

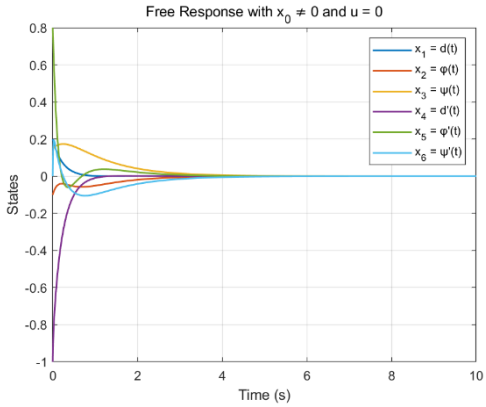


Fig. 3-2. States of Free Response

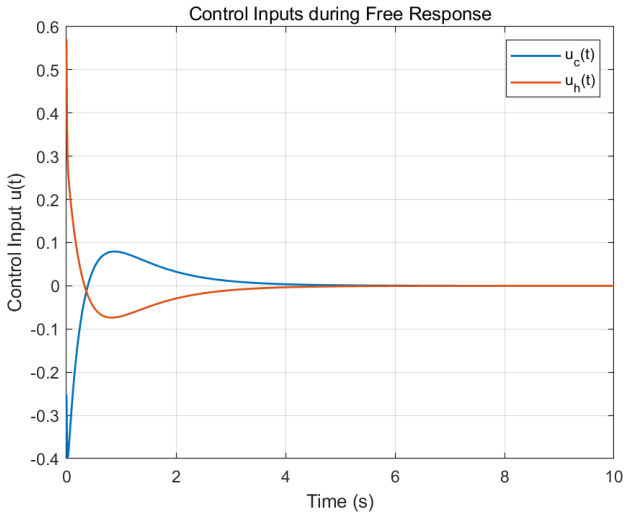


Fig. 3-3. Control of Free Response

The real-time output responses under step inputs $r = [0,1]$ and $[1,0]$, with an initial state of 0 are also illustrated in the figures below.

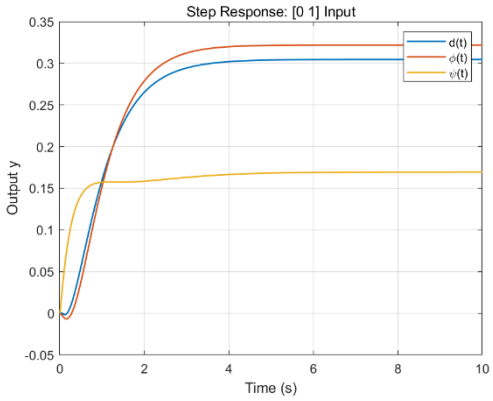


Fig. 3-4. Output of Step Response [0,1]

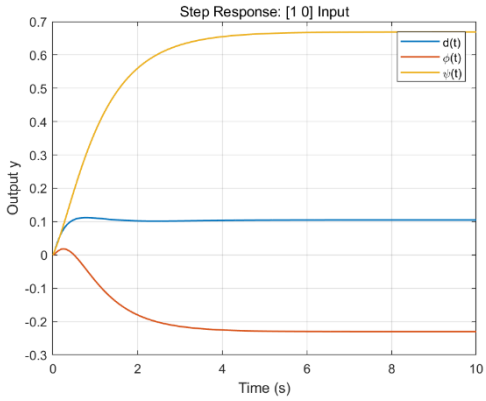


Fig. 3-5. Output of Step Response [1,0]

The program output obtained by running the MATLAB script *task2.m* is shown below.

```
Max control input during free response:
u_c: 0.40
u_h: 0.57

=== Performance Report for Input [1, 0] ===
Output: d(t)   | Overshoot:    6.72% | Settling Time:  3.53s
Output: phi(t) | Overshoot:    0.00% | Settling Time:  4.12s
Output: psi(t) | Overshoot:    0.00% | Settling Time:  4.07s

=== Performance Report for Input [0, 1] ===
Output: d(t)   | Overshoot:    0.00% | Settling Time:  3.40s
Output: phi(t) | Overshoot:    0.00% | Settling Time:  3.26s
Output: psi(t) | Overshoot:    0.00% | Settling Time:  3.85s
```

Based on the output above and Figure 3-4 and 3-5, it can be concluded that the pole placement fully meets the transient response requirements—namely, the overshoot is less than 10% and the settling time is less than 5 seconds. The max control inputs during free response are $u_c = 0.40$ and $u_h = 0.57$ respectively. The size of the control signals is reasonable.

I also built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task2.m*. The block diagrams corresponding to the free response and step response are shown below.

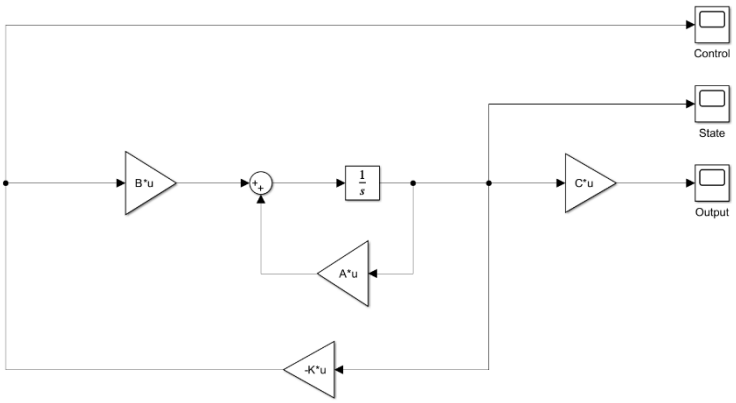


Fig. 3-6. Model of Free Response Using LQR Method

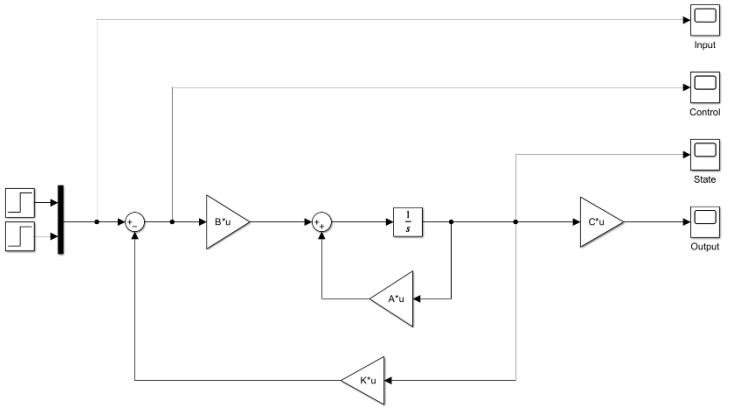


Fig. 3-7. Model of Step Response Using LQR Method

The corresponding observation results by the scopes are shown in the following figures.

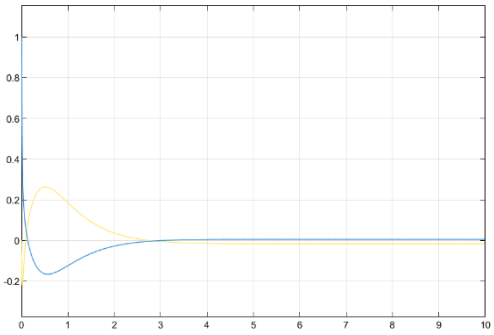


Fig. 3-8. Control of Step Response [0,1] (SIM)

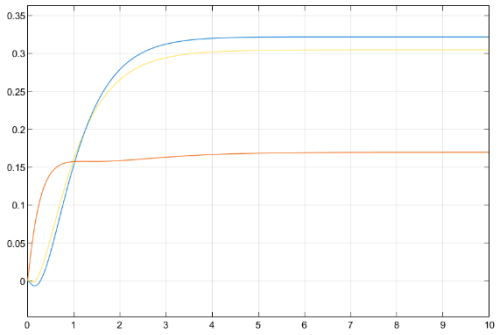


Fig. 3-9. Output of Step Response [0,1] (SIM)

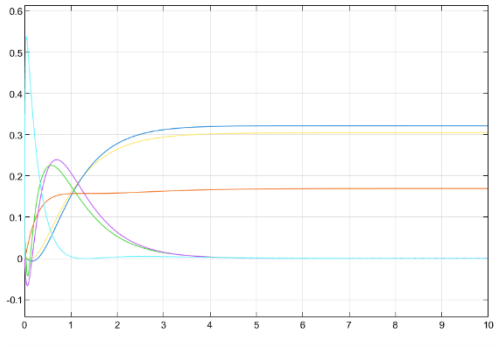


Fig. 3-10. States of Step Response [0,1] (SIM)

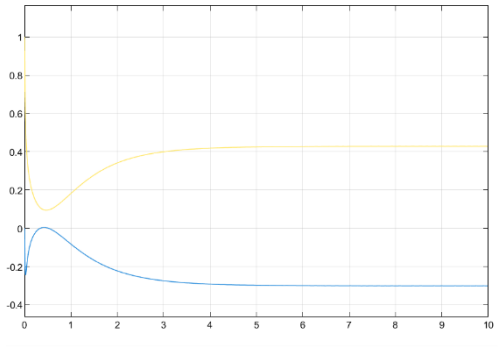


Fig. 3-11. Control of Step Response [1,0] (SIM)

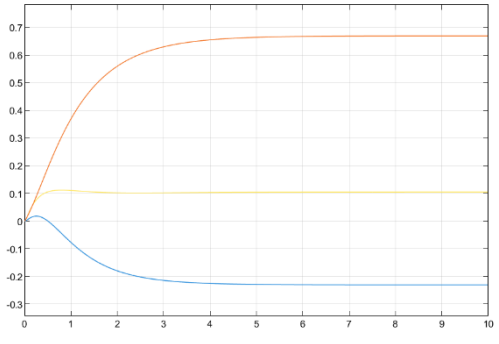


Fig. 3-12. Output of Step Response [1,0] (SIM)

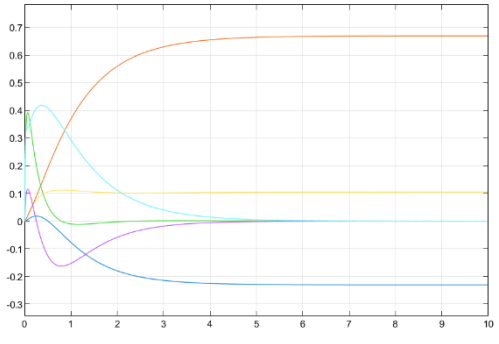


Fig. 3-13. States of Step Response [1,0] (SIM)

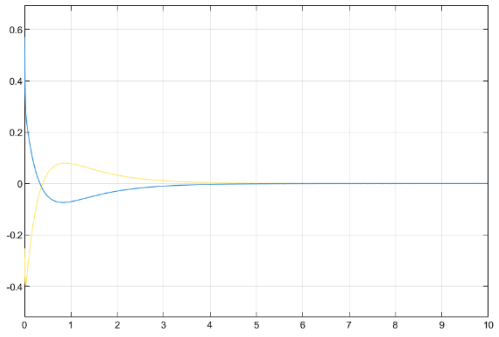


Fig. 3-14. Control of Step Response [0,1] (SIM)

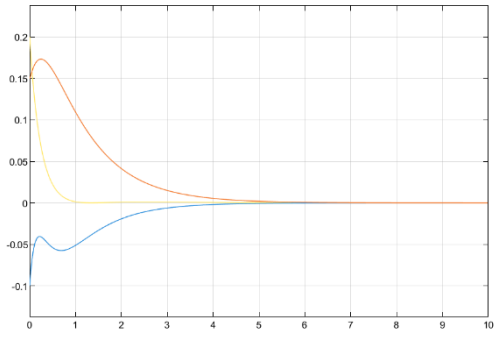


Fig. 3-15. Output of Free Response (SIM)

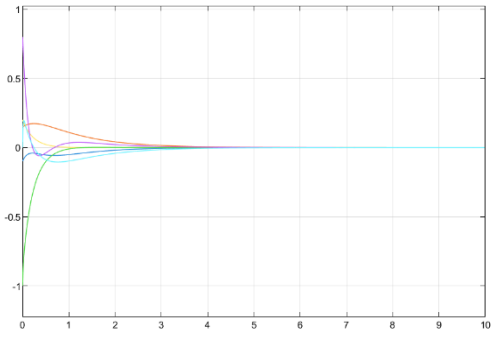


Fig. 2-16. States of Free Response (SIM)

As can be seen by comparing with the figures above, the results obtained from running the MATLAB script *task2.m* are fully consistent with those obtained from the scopes in the SIMULINK simulation. This confirms that both the implementation of the code and the structure of the SIMULINK model diagram are correct.

According to prior knowledge, Q is the state weighting matrix, where each element on the diagonal corresponds to a state of the system. R is the control weighting matrix, with each diagonal element corresponding to one control input of the system. Therefore, I will mainly compare the changes in system states and control signals under different Q and R matrices.

Firstly, we modify a specific element in Q. For example, changing the third diagonal element from 5 to 50 — the system's states under free response are as follows

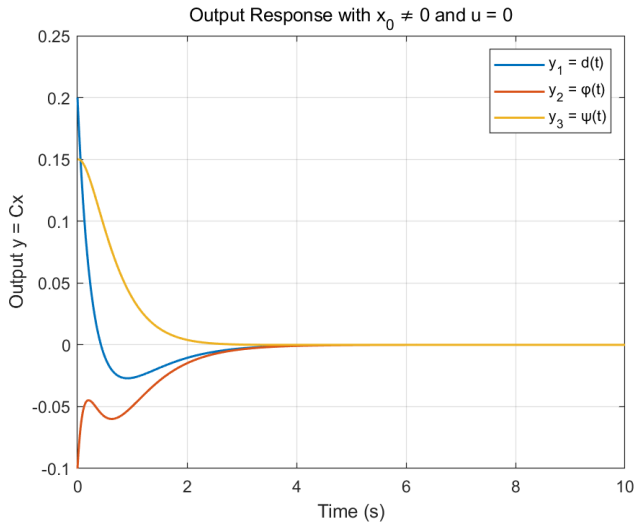


Fig. 2-17. Output of Free Response ($Q_{33} = 50$)

If the same element is changed from 5 to 0.1, the system's states under free response become

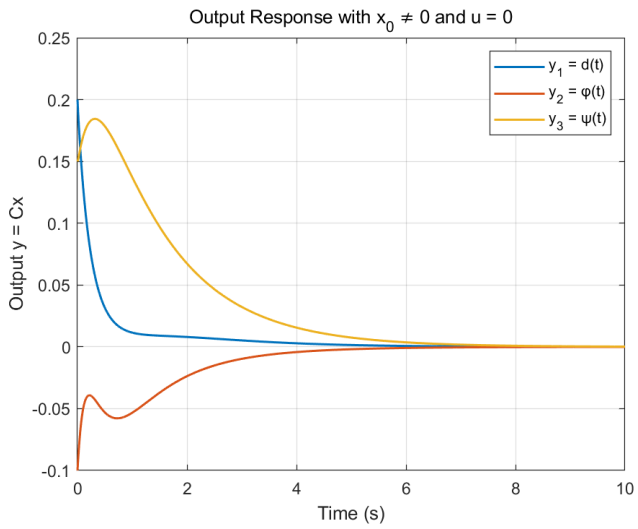


Fig. 2-18. Output of Free Response ($Q_{33} = 0.1$)

Next, by increasing both diagonal elements of the R matrix from 5 to 50, the system's control inputs under step response $[1, -0]$ behave as follow

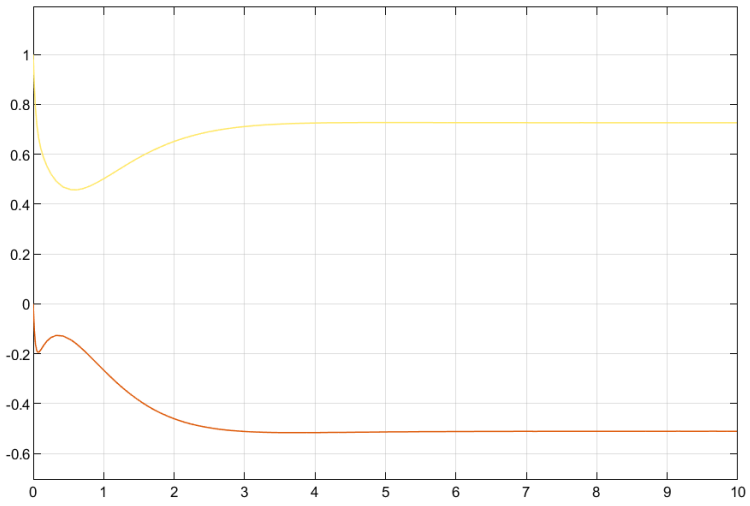


Fig. 2-19. Control of Step Response [1,0] ($R = diag([50,50])$)

Conversely, if both diagonal elements of R are reduced from 5 to 0.5, the system's control inputs under step response $[1, -0]$ become

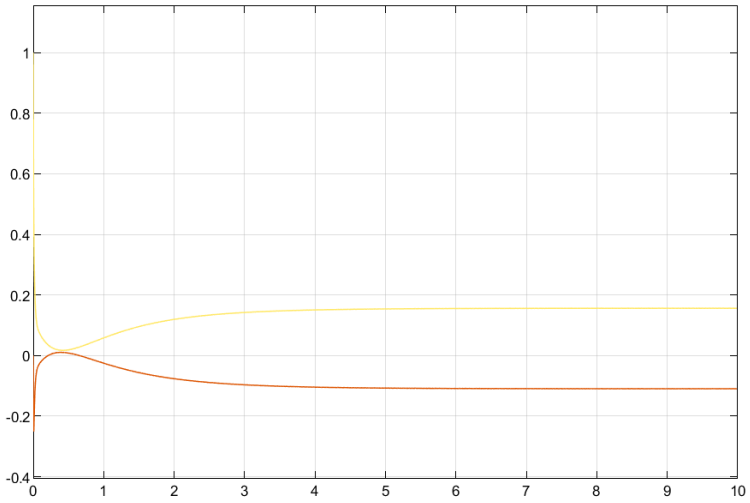


Fig. 2-20. Control of Step Response [1,0] ($R = diag([0.5,0.5])$)

Based on the results above, it can be confirmed that Q as the state weighting matrix, determines the extent to which deviations in the system states from their desired values are penalized during optimal control. If certain diagonal elements of Q are large, the system places more emphasis on minimizing the deviations of the corresponding state variables, resulting in faster convergence to steady-state. Conversely, smaller values lead to slower responses.

On the other hand, R as the control weighting matrix, determines how much the control effort is penalized. If certain diagonal elements of R are large, it indicates that using control inputs incurs higher cost, and the system will attempt to limit the magnitude of changes in those control signals, avoiding excessive variations.

4 Observer-Based LQR Control System Design

4.1 Design Methodology

This task requires to design a full-order state observer by using the dual system pole placement method to compute the gain matrix L, enabling estimation of the system's internal states. It is necessary to check if the system is observable. We first construct an observability matrix as

$$W_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix} \tag{32}$$

The rank of W_o is 6, which means the system is observable. Next, I continue to use

the LQR method to design the controller. Since the Q and R matrices used in Task 2 have already been validated against the design specifications, I reuse the same Q and R to ensure the system meets the transient response requirements. The state feedback gain matrix K is then obtained using the same approach by solving the Algebraic Riccati Equation.

I aim to build a full-order observer, so consider an estimator

$$\begin{aligned} \hat{x}'(t) &= A\hat{x}(t) + Bu(t) + L[y - \hat{y}] \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{33}$$

For this dual system

$$\tilde{A} = A^T, \quad \tilde{B} = C^T, \quad \tilde{K} = L^T \tag{34}$$

I still follow the same approach as in Task 1: first, I construct the structured controllability basis and the transformation matrix T; then I design the desired poles and form the characteristic polynomials. The system is transformed into the controllable canonical form, and pole placement is performed. As mentioned in Chapter 11, I placed observer poles 3-5 times faster than the closed loop dominant poles designed by the controlled. The desired poles are as shown in Table 4-1

p_1	p_2	p_3	p_4	p_5	p_6
-10	-10	-20	-20	-30	-30

TABLE 4-1. Pole Placement

Finally, the observer gain matrix $\widetilde{K}$ is obtained, hence L.

$$L = \begin{bmatrix} 2 & 0 & 0 \\ -3.8733 & 35.7978 & 0.3149 \\ 0 & 0 & 48.8636 \\ 64 & 6.5 & -10 \\ -12.9885 & 267.5945 & 19.1482 \\ 5 & -3.6 & 355.8368 \end{bmatrix} \tag{35}$$

4.2 Results and Analysis

I built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task3.m*. The block diagram corresponding to the free response is shown below.

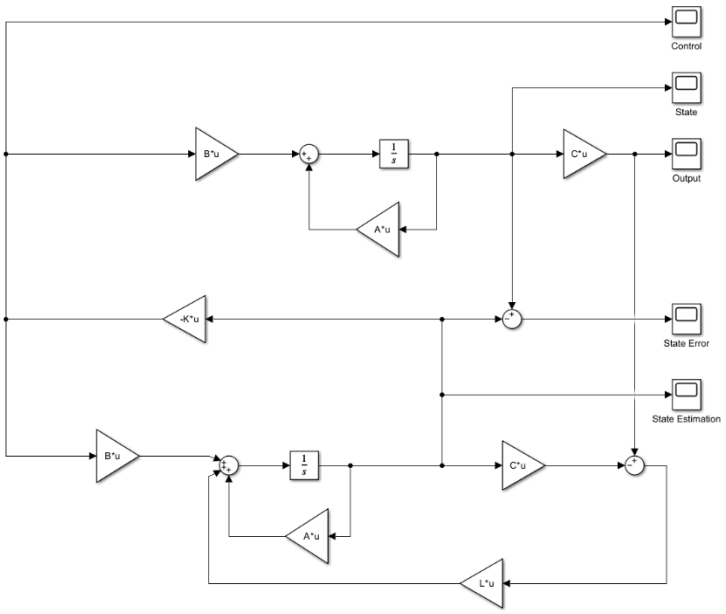


Fig. 4-1. Model of Free Response of Observer-Based LQR Controller System

The states, output, and control signals under free response obtained through SIMULINK simulation, as well as the estimated states and state estimation errors obtained from the observer, are shown in the figures below.

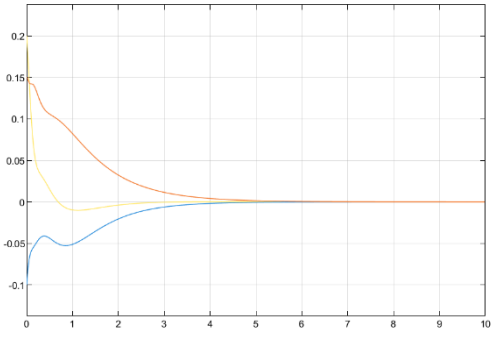


Fig. 4-2. Output of Free Response (SIM)

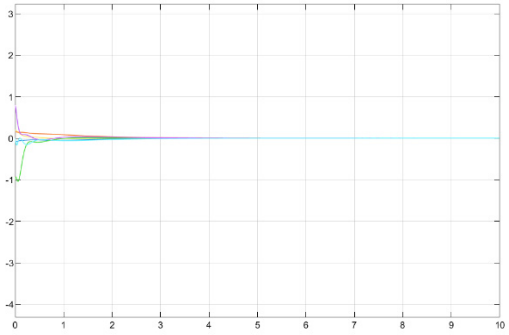


Fig. 4-3. States of Free Response (SIM)

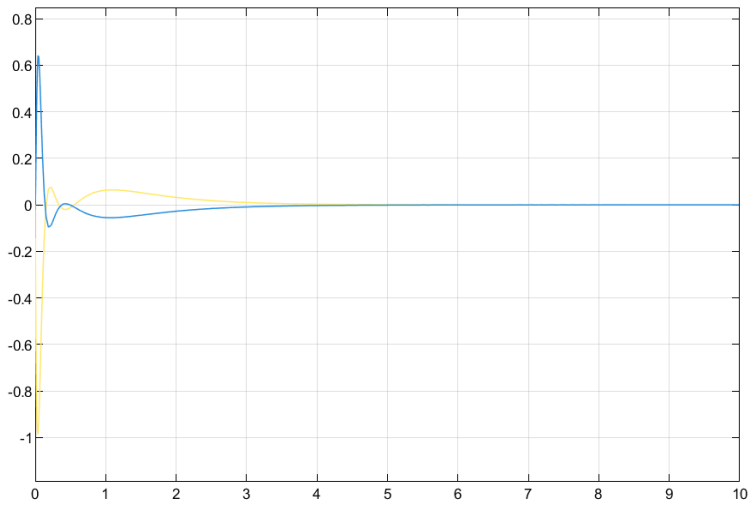


Fig. 4-4. Control of Free Response (SIM)

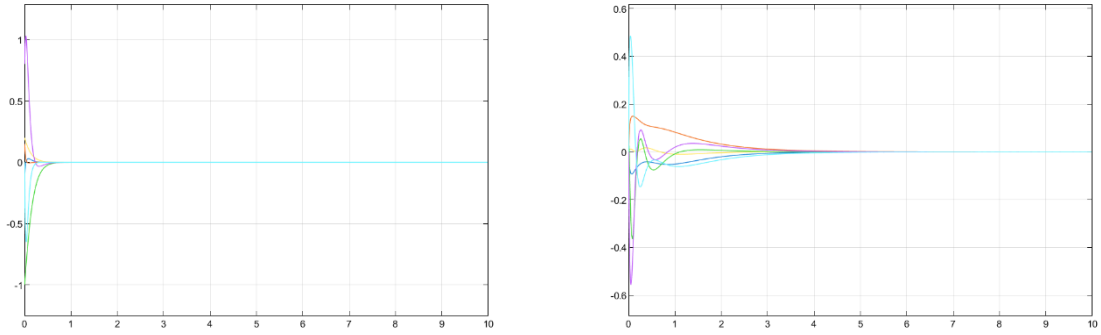


Fig. 4-5/6. States Estimation Error and States Estimation of Free Response (SIM)

Firstly, verification shows that the designed observer-based LQR control system meets the transient performance specifications of the system. Secondly, when compared to the state and output of free response of the original system obtained in Task 2, it can be observed that the system with the observer exhibits some state estimation error, particularly at $t = 0$ and shortly afterward. However, due to the presence of closed-loop feedback, this error quickly converges to zero and the system reaches steady state. This indicates that the observer is capable of accurately estimating the real-time states of the system and the design meets the required specifications

To investigate effects of observer poles on state estimation error and closed-loop control performance, I modified the poles used in the observer design, with the specific changes shown in Table 4-2. The original pole configuration is $p = [-10, -10, -20, -20, -30, -30]$.

Configuration	p_1	p_2	p_3	p_4	p_5	p_6
1	-5	-5	-10	-10	-15	-15

2	-30	-30	-40	-40	-50	-50

TABLE 4-2. Pole Placement for 2 Different Configurations

The system state estimation errors and actual states obtained by the observer under different pole configurations are shown in the figures below.

Configuration 1:

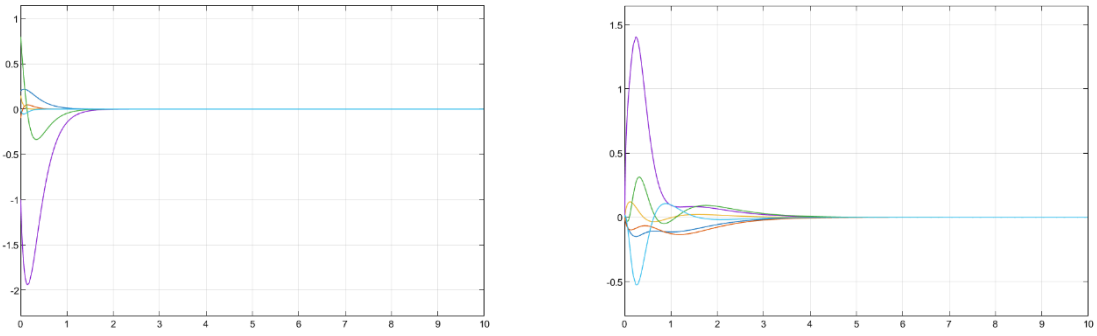


Fig. 4-7/8. States Estimation Error and States Estimation of Free Response (SIM) (Config. 1)

Configuration 2:

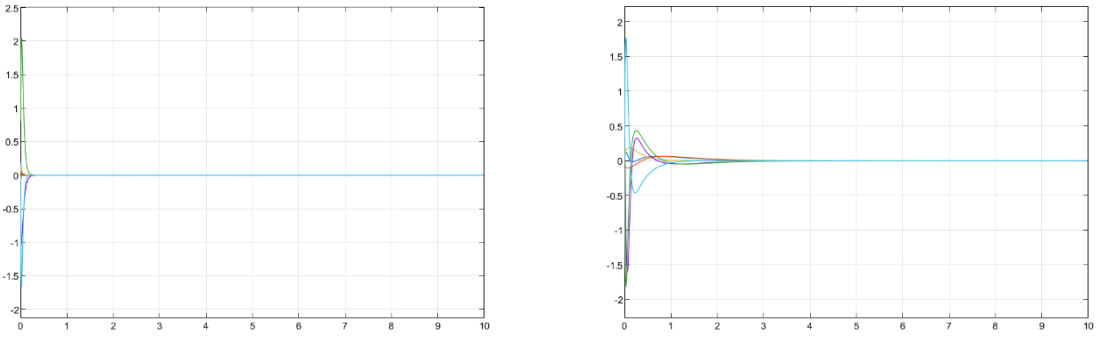


Fig. 4-9/10. States Estimation Error and States Estimation of Free Response (SIM) (Config. 2)

From the figures above, it can be observed that if the poles of the observer are closer to the imaginary axis, the system states exhibit more significant overshoot and require a longer time to reach steady state. Additionally, the error between the estimated states and the true states becomes larger. On the contrary, if the observer poles are more negative, the system states have smaller overshoot, settle faster, and the estimation error between the estimated and actual states is reduced.

5 Decoupling Controller Design

5.1 Design Methodology

This task involves designing a decoupling control system. We need to construct a state

feedback controller such that the system can be decoupled into two independent subsystems, enabling each output to respond only to its designated input channel.

We are only interested in the two outputs $d(t)$ and $\psi(t)$, so the output matrix C should be C_2, then we get a 2-input-2output system.

$$C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{36}$$

The closed loop system is

$$\begin{aligned} \dot{x} &= (A - BK)x + BFr \\ y &= C_2 x \end{aligned} \tag{37}$$

and the transfer function matrix of the feedback system is

$$H(s) = G(s)[I + K(sI - A)^{-1}B]^{-1}F \tag{38}$$

where

$$G(s) = C(sI - A)^{-1}B \tag{39}$$

The next step is to compute the relative degree σ_i, which helps determine from which order each output begins to respond to the input. This is essential for preparing the construction of the decoupling controller.

$$\sigma_i = \begin{cases} \min\left(j \middle| c_i^T A^{j-1} B \neq 0^T, j = 1,2,\dots,n\right) \\ n, \ \ if\ c_i^T A^{j-1} B \neq 0^T, j = 1,2,\dots,n \end{cases} \tag{40}$$

In this system, both the outputs have the relative degree of 2. Then we need to construct the leading coefficient matrix B^* as shown below.

$$B^* = \begin{bmatrix} c_1^T A^{\sigma_1 - 1} B \\ c_2^T A^{\sigma_2 - 1} B \end{bmatrix} = \begin{bmatrix} 27 & 11.2 \\ 40 & 61.4 \end{bmatrix} \tag{41}$$

Since both relative degrees are 2, I assigned two desired poles for each output channel, namely $[-6, -8]$ and $[-10, -12]$. Then, I constructed the corresponding stable characteristic polynomials for the two output channels.

$$\begin{aligned} \phi_{f1}(s) &= (s + 6)(s + 8) = s^2 + 14s + 48 \\ \phi_{f2}(s) &= (s + 10)(s + 12) = s^2 + 22s + 120 \end{aligned} \tag{42}$$

By substituting s with A, the resulting matrix functions were used to construct the zero polynomial coefficient matrix C^* as shown below.

$$C^* = \begin{bmatrix} c_1^T \phi_{f1}(A) \\ c_2^T \phi_{f2}(A) \end{bmatrix} = \begin{bmatrix} 48 & 6.5 & -10 & -4 & 0 & 0 \\ 5 & -3.6 & 120 & 0 & 0 & 10.8636 \end{bmatrix} \tag{43}$$

As described in Chapter 10, the feedforward compensation matrix F and the state

feedback gain matrix K are computed as follows.

$$F = (B^*)^{-1} = \begin{bmatrix} 0.0508 & -0.0093 \\ -0.0331 & 0.0223 \end{bmatrix} \tag{44}$$

$$K = (B^*)^{-1}C^* = \begin{bmatrix} 2.3898 & 0.3632 & -1.6184 & -0.2030 & 0 & -0.1006 \\ -1.4755 & -0.2953 & 3.0088 & 0.1323 & 0 & 0.2425 \end{bmatrix} \tag{45}$$

5.2 Results and Analysis

The real-time variations of the output, state, and control signals under the system's free response—starting from the initial state x_0 with no external input—are shown in the figures below.

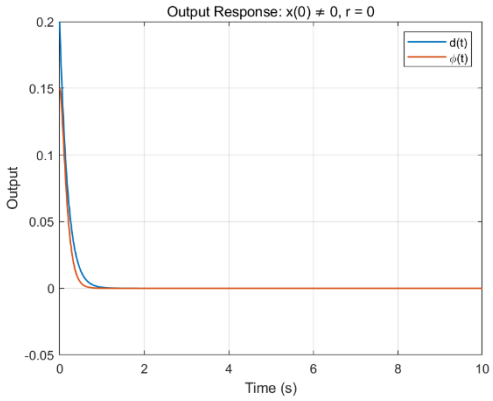


Fig. 5-1. Output of Free Response

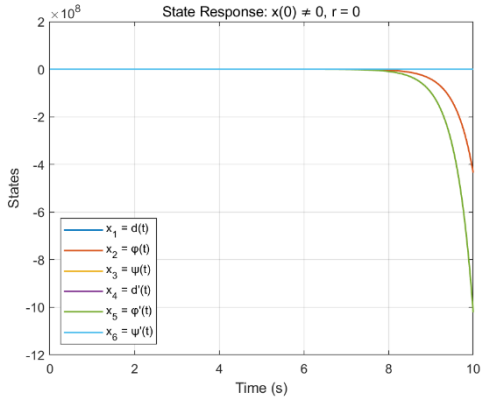


Fig. 5-2. States of Free Response

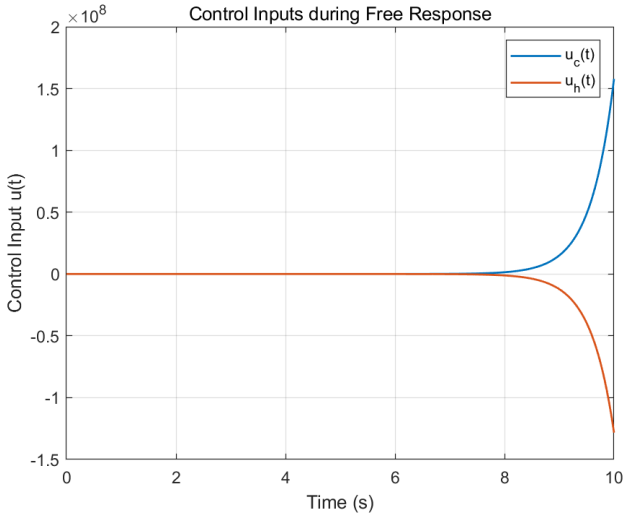


Fig. 5-3. Control of Free Response

The real-time output responses under step inputs $r = [0,1]$ and $[1,0]$, with an initial state of 0 are also illustrated in the figures below.

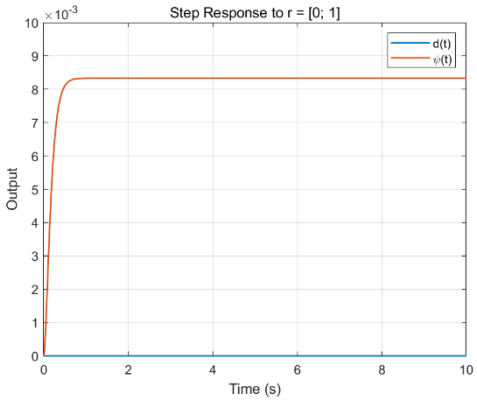


Fig. 5-4. Output of Step Response [0,1]

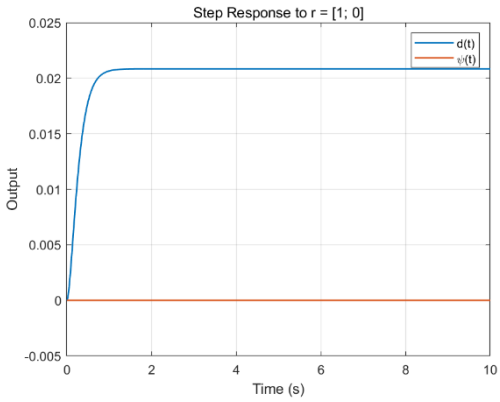


Fig. 5-5. Output of Step Response [1,0]

From the output results of the step responses, it can be observed that when $r = [0,1]$, $\psi(t)$ responds rapidly while $d(t)$ remains unchanged. When $r = [1,0]$, $d(t)$ responds rapidly while $\psi(t)$ remains unchanged. This demonstrates that the system is successfully decoupled and that the decoupling performance is stable.

However, under free response with the initial condition x_0 and no input, the variations in the state and control signals indicate that the decoupled system is not internally stable.

I also built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task4.m*. The block diagrams corresponding to the free response and step response are shown below.

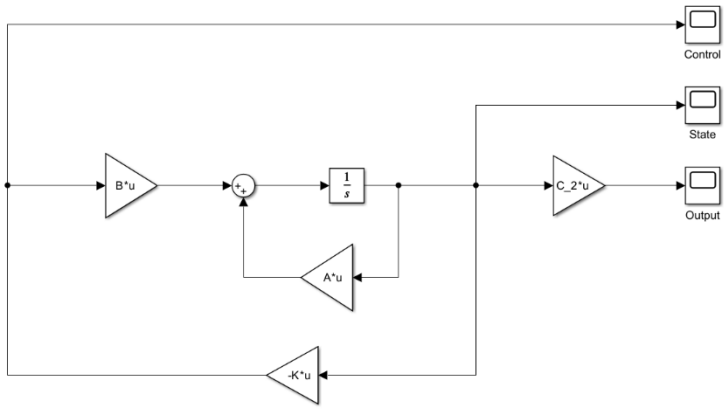


Fig. 5-6. Model of Free Response of the Decoupling Controller System

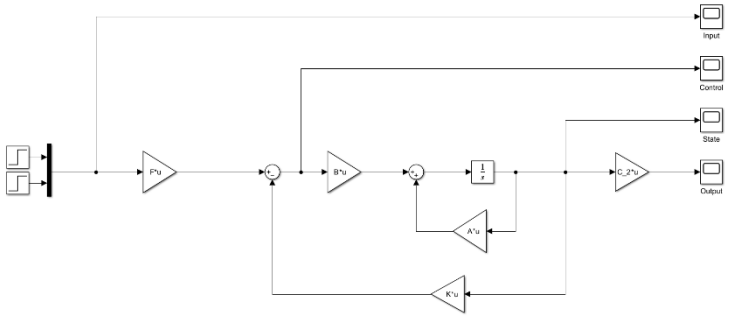


Fig. 5-7. Model of Step Response of the Decoupling Controller System

The corresponding observation results by the scopes are shown in the following figures.

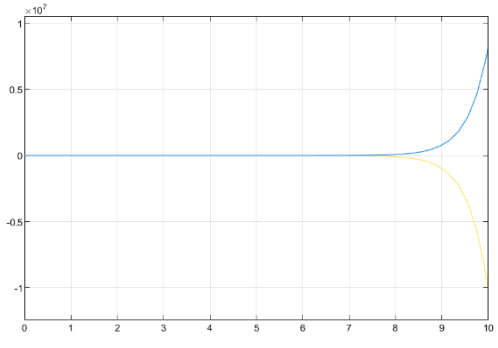


Fig. 5-8. Control of Step Response [0,1] (SIM)

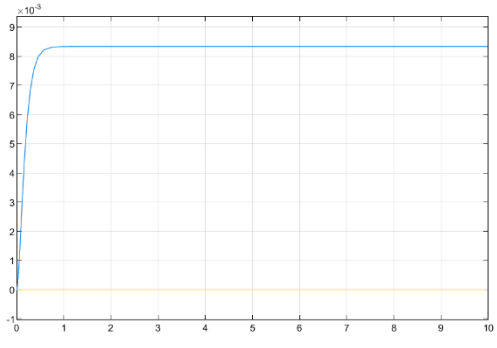


Fig. 5-9. Output of Step Response [0,1] (SIM)

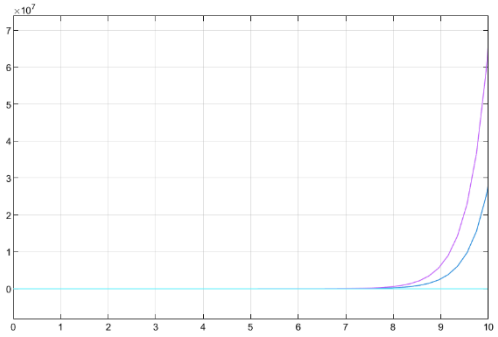


Fig. 5-10. States of Step Response [0,1] (SIM)

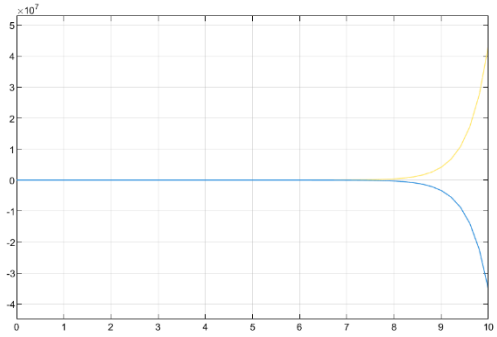


Fig. 5-11. Control of Step Response [1,0] (SIM)

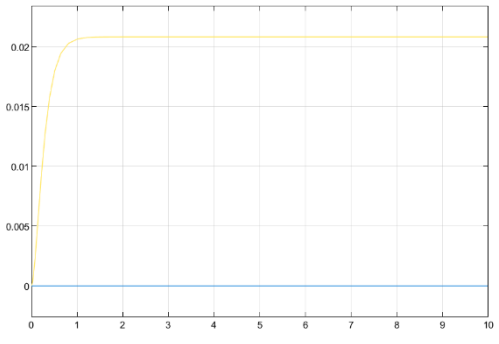


Fig. 5-12. Output of Step Response [1,0] (SIM)

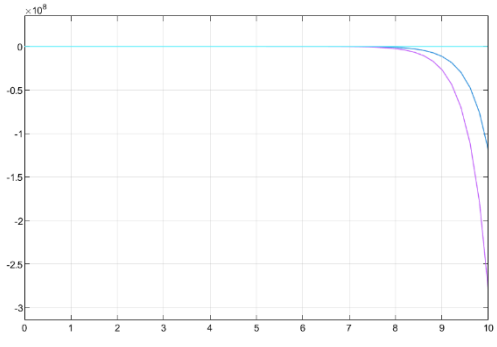


Fig. 5-13. States of Step Response [1,0] (SIM)

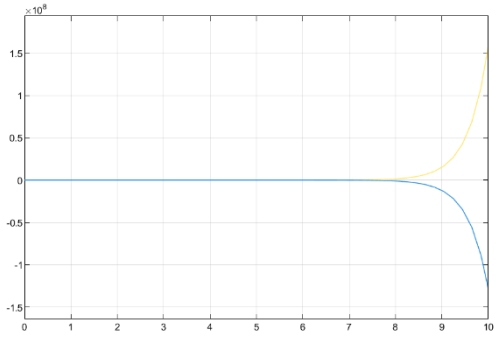


Fig. 5-14. Control of Step Response [0,1] (SIM)

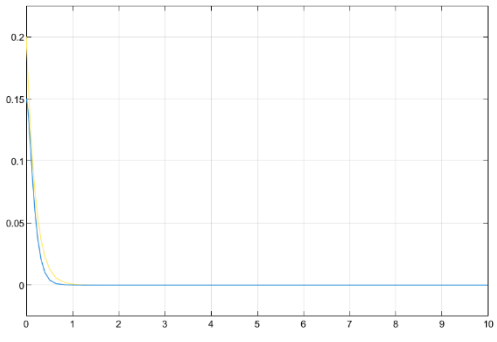


Fig. 5-15. Output of Free Response (SIM)

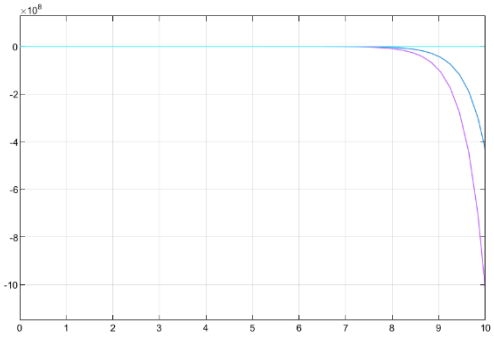


Fig. 5-16. States of Free Response (SIM)

As can be seen by comparing with the figures above, the results obtained from running the MATLAB script *task4.m* are fully consistent with those obtained from the scopes in the SIMULINK simulation. This confirms that both the implementation of the code and the structure of the SIMULINK model diagram are correct.

6 Integral Controller Design

6.1 Design Methodology

My matriculation number is "A0313771H". Then y_{sp} is calculated based on that.

$$y_{sp} = -\frac{1}{10} C A^{-1} B \begin{bmatrix} -0.5 + \dfrac{a-b}{20} \\ 0.1 + \dfrac{b-c}{a+d+10} \end{bmatrix} = \begin{bmatrix} 1.7618 \\ 1.8397 \\ 1.0180 \end{bmatrix} \tag{46}$$

To achieve zero steady state error, we introduce one integrator to each channel

$$v(t) = \int_0^t e(\tau)\, d\tau \tag{47}$$

Then it follows that

$$\dot{v}(t) = e(t) = r - y(t) = r - Cx(t) \tag{48}$$

We can then construct an augmented system

$$\begin{aligned} &\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} B \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ I \end{bmatrix} r \\ &\bar{x}' = \bar{A}\bar{x} + \bar{B}u + \bar{B}w + \overline{B_r} r \\ &y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \end{aligned} \tag{49}$$

The next step is to check whether the augmented system is controllable or not:

$$rank \begin{pmatrix} A & B \\ -C & 0 \end{pmatrix} = n + m = 8 \tag{50}$$

So, the augmented system is controllable. Since the same Q and R are the same with those used in Task 2, the original system is controllable and meets the design specifications, too. Then by using LQR method, the gain matrix K can be calculated.

$$K = \begin{bmatrix} 20.1088 & -17.4848 & -5.1803 & 2.0257 & -2.5925 & 0.1813 & 0.0111 & 1.5823 & -3.5582 \\ -21.6819 & 21.4473 & 11.0248 & -1.6532 & 3.2596 & 1.6005 & -0.4452 & -2.9210 & -1.7430 \end{bmatrix} \tag{51}$$

For the augmented system, I still used LQR Method to build the observer gain matrix L, where $Q = eyes(6)$ and $R = eye(3)$.

$$L = \begin{bmatrix} 0.9895 & -0.2744 & -0.1093 \\ -0.2744 & 5.5974 & -0.1394 \\ -0.1093 & -0.1394 & 1.0173 \\ 0.0332 & 1.7695 & -0.5949 \\ -3.5615 & 15.2126 & 0.7073 \\ 0.4137 & -1.5997 & 0.0331 \end{bmatrix} \tag{52}$$

The corresponding observer is in the form of

$$\hat{x}' = A\hat{x} + Bu + L(y - \hat{y}) \tag{53}$$

6.2 Results and Analysis

I built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task5.m*. The block diagram corresponding to the free response is shown below.

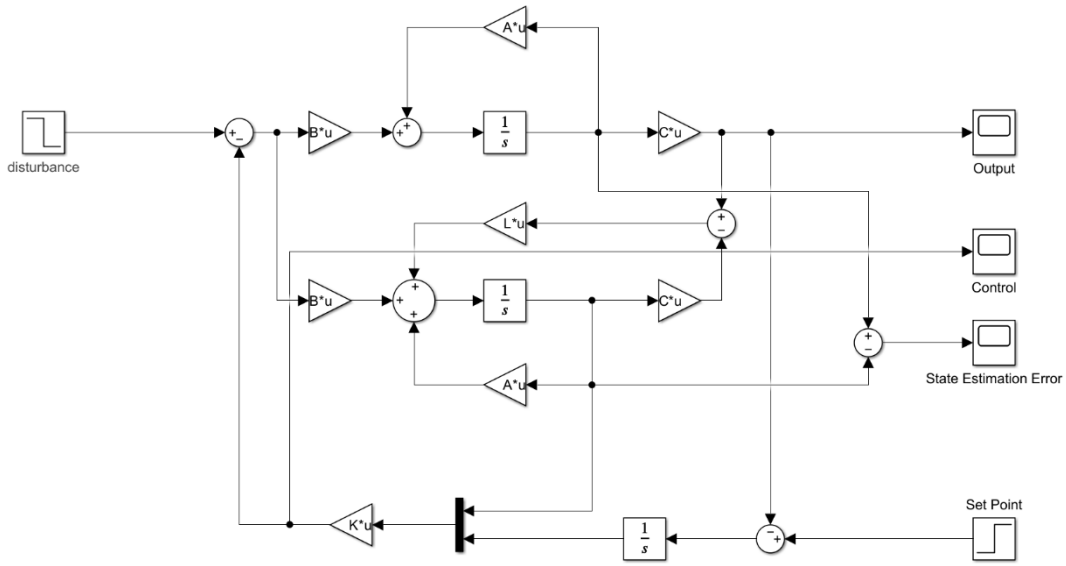


Fig. 6-1. Model of Multivariable Integral Control

The corresponding observation results by the scopes are shown in the following figures.

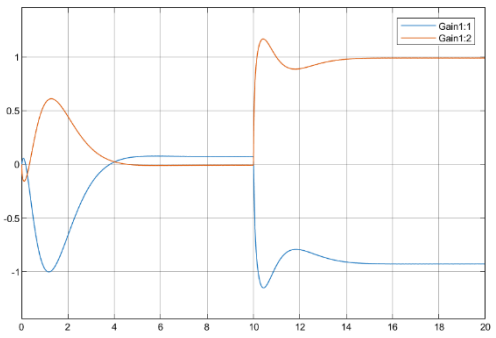


Fig. 6-2. Control of Integral Control

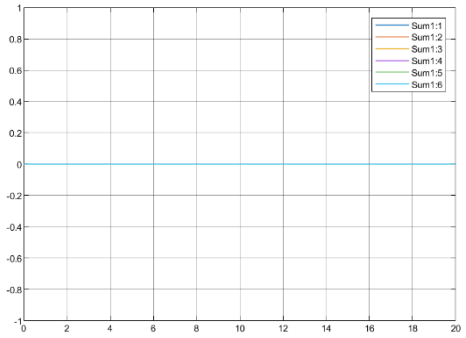


Fig. 6-3. States Estimation Error of Integral Control

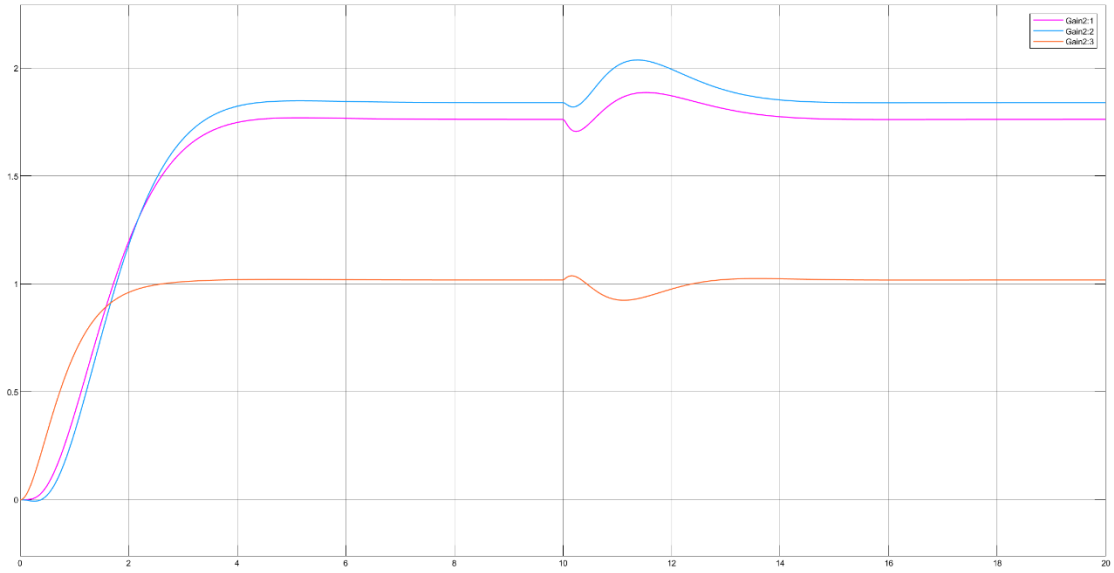


Fig. 6-4. Output of Integral Control

As can be seen from the figures above, the controller maintains the three outputs at any arbitrary set point. The system rejects constant disturbances at steady state due to integrator action.

7 Discussion on Arbitrary Constant Set points

7.1 Mathematical Analysis

To achieve zero steady state error, we need to construct the system following

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} B \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ I \end{bmatrix} r \tag{54}$$

To ensure that the system achieves zero steady-state error for any constant reference y_{sp}, the closed-loop system must be stable. And the augmented system must be controllable. According to classical multivariable control theory, a necessary condition to track an arbitrary constant reference is

$$rank \begin{pmatrix} A & B \\ -C & 0 \end{pmatrix} = n + p \tag{55}$$

where n is the number of states, p is the number of outputs, m is the number of inputs. If this rank condition is not satisfied, perfect tracking for any arbitrary reference is mathematically impossible, even if the system is controllable and stable.

In our case, $n = 6$, $p = 3$ and $m = 2$, we compute

$$rank \begin{pmatrix} A & B \\ -C & 0 \end{pmatrix} = n + p = 8 < 9 \tag{56}$$

This means the system does not have enough inputs to independently control all 3 outputs at arbitrary values. Therefore, We cannot guarantee zero steady-state error for arbitrary constant set points, although tracking some specific set points is still feasible.

7.2 Results

I built the system's model block diagram using SIMULINK, and performed simulations based on the parameters computed in *task6.m*. The block diagram corresponding to the free response is shown below.

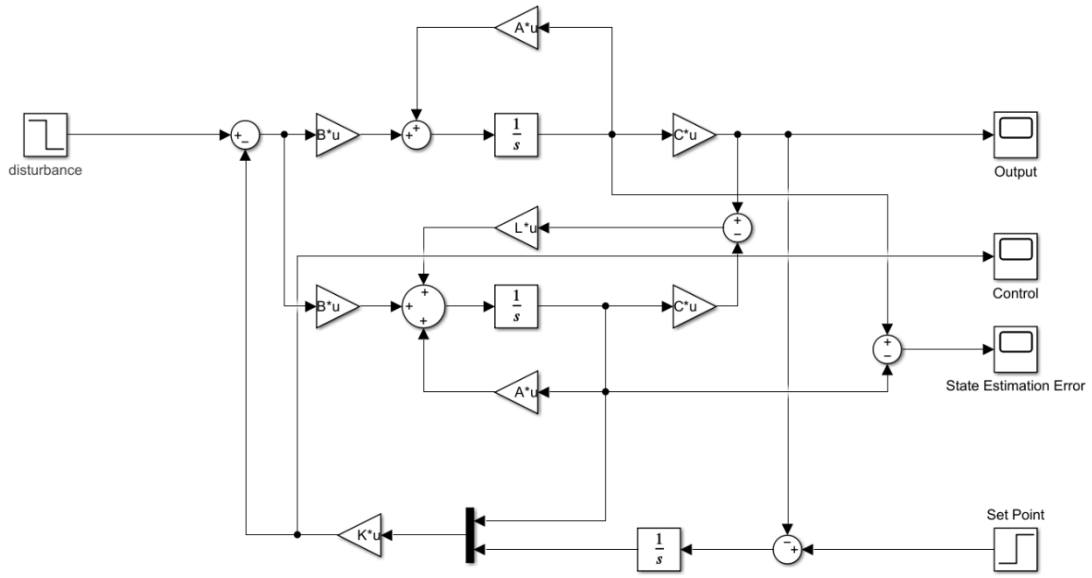


Fig. 7-1. Model of Multivariable Integral Control

I set the set point y_{sp} to $[1,2,3]$ and $[10,20,30]$ respectively. The system's outputs are shown in the following two figures. As we can see, although the system still tries to maintain a steady state after the disturbance is introduced, the outputs are unable to reach the desired set points. That is, there remains a significant error between the target values and the actual values.

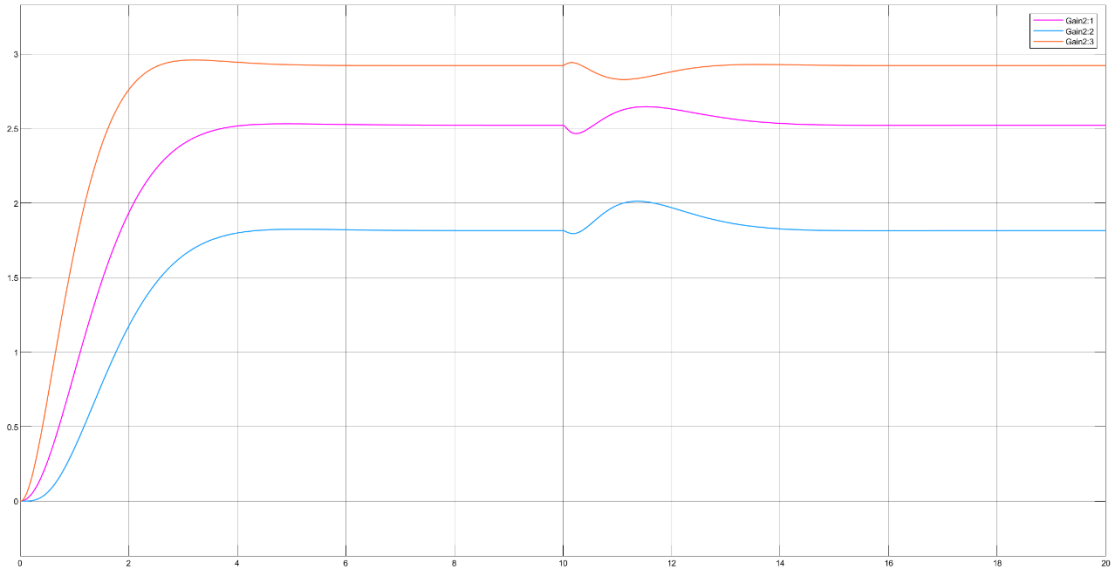


Fig. 7-2. Output of Integral Control ($y_{sp} = [1,2,3]$)

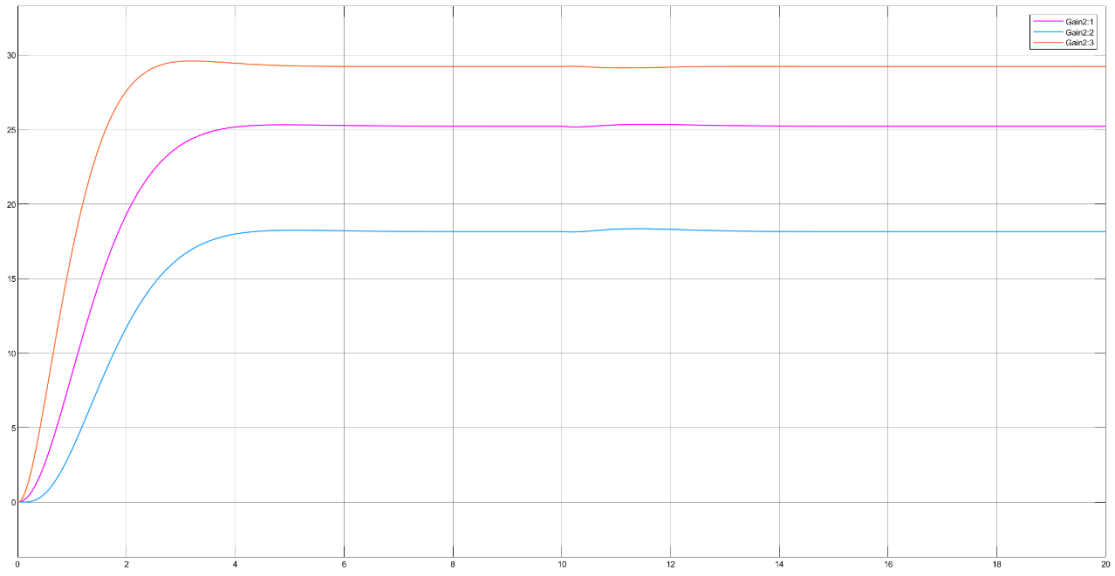


Fig. 7-3. Output of Integral Control ($y_{sp} = [10,20,30]$)

8 Conclusion

This project aims to design and evaluate advanced control strategies for a stationary self-balancing two-wheeled vehicle system using state-space methods. Six progressive tasks are implemented, including pole placement control, LQR design, observer-based control, servo control with disturbance rejection, and decoupling control.

Throughout the tasks, several classical and modern control techniques were implemented, such as pole placement via controllable canonical form, Linear Quadratic Regulator, full-state observer, and integral servo control. Most of the controllers are verified through both MATLAB-based simulation and Simulink block diagram models, ensuring accurate reproduction of system dynamics.

Through the completion of this project, I have significantly deepened my understanding of modern control theory, particularly in state-space analysis, pole placement, and LQR-based optimization. The project also highlighted the importance of system controllability and observability in ensuring successful controller design.

Appendix

parameter_init.m

```
function output = parameter_init()
    % Metriculation Number: A0313771H
    a = 3; b = 7; c = 7; d = 1;

    g = 9.8; % Gravitational acceleration

    M_f = 2.14 + c/20; % Mass of the front wheel
    M_r = 5.91 - b/10; % Mass of the rear wheel
    M_c = 1.74; % Mass of the cart system

    L_F = 0.133; % Horizontal length from a front wheel rotation axis to
the steering axis
    L_Ff = 0.05; % Horizontal length from a front wheel rotation axis to
a center-of-gravity of part of front wheel
    L_R = 0.308 + (a - d)/100; % Horizontal length from a rear wheel
rotation axis to the steering axis
    L_r = 0.128; % Horizontal length from a rear wheel rotation axis to
a center-of-gravity of part of rear wheel
    L_c = 0.259; % Horizontal length from a rear wheel rotation axis to
a center-of-gravity of the cart system

    J_x = 0.5 + (c - d)/100; % Moment of inertia around center-of-
gravity x axially

    H_f = 0.18; % Vertical length from a floor to a center-of-gravity of
the front wheel
    H_r = 0.161; % Vertical length from a floor to a center-of-gravity
of the rear wheel
    H_c = 0.098; % Vertical length from a floor to a center-of-gravity
of the front wheel

    mu_x = 3.33 - b/20 + a*c/60; % Viscous coefficient around x axis

    alpha = 15.5 - a/3 + b/2;
    gamma = 11.5+ (a - c)/(b + d + 3);
    beta = 27.5 - d/2;
    delta = 60 + (a - d)*c/10;
```

```
    %% Parameters in the matrices
    den = M_f*H_f^2 + M_r*H_r^2 + M_c*H_c^2 + J_x;
    a_51 = -M_c*g/den;
    a_52 = (M_f*H_f + M_r*H_r + M_c*H_c)*g/den;
    a_53 = (M_r*L_r*L_F + M_c*L_c*L_F + M_f*L_Ff*L_R)*g/((L_R+L_F)*den);
    a_54 = -M_c*H_c*alpha/den;
    a_55 = -mu_x/den;
    a_56 = M_f*H_f*L_Ff*gamma/den;
    b_51 = M_c*H_c*beta/den;
    b_52 = -M_f*H_f*L_Ff*delta/den;

    A = [0, 0, 0, 1, 0, 0;
         0, 0, 0, 0, 1, 0;
         0, 0, 0, 0, 0, 1;
         0, 6.5, -10, -alpha, 0, 0;
         a_51, a_52, a_53, a_54, a_55, a_56;
         5, -3.6, 0, 0, 0, -gamma];

    B = [0, 0;
         0, 0;
         0, 0;
         beta, 11.2;
         b_51, b_52;
         40, delta];

    C = [1, 0, 0, 0, 0, 0;
         0, 1, 0, 0, 0, 0;
         0, 0, 1, 0, 0, 0];

    y_sp = -0.1*C/A*B*[-0.5 + (a - b)/20; 0.1 + (b - c)/(a + d + 10)];

    x_0 = [0.2; -0.1; 0.15; -1; 0.8; 0];

    output = {A, B, C, y_sp, x_0};
end
```

task1.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
```

```
syms s;
syms k_11_bar k_12_bar k_13_bar k_14_bar k_15_bar k_16_bar k_21_bar
k_22_bar k_23_bar k_24_bar k_25_bar k_26_bar;
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
x_0 = output{5};

%% === Design desired poles ===
zeta = 0.8;
omega = 1.5; % (ζ = 0.8, ωn = 1.5)
p_dominant = [-zeta*omega + omega*sqrt(1-zeta^2)*1j, -zeta*omega -
omega*sqrt(1-zeta^2)*1j]; % Dominant poles
p_extra = [-4, -4, -6, -6]; % Extra non-dominant poles (2~5 times for
faster convergence)
desired_poles = [p_dominant, p_extra]; % Get all the poles
% K_ideal = place(A, B, desired_poles); % Get the ideal feedback gain
matrix K to place the desired poles on closed-loop system

max_ost = 0.1; % The overshoot is less than 10%
max_st = 5; % The 2% settling time is less than 5 seconds
M_p = exp(-pi*zeta/sqrt(1-zeta^2)); % Ideal overshoot calculation
t_s = 4/(zeta*omega); % Ideal settling time calculation

fprintf("According to the ideal second-order system formula\nThe
overshoot is %.2f%%.\nThe 2%% settling time is %.3f seconds.\n",
M_p*100, t_s);

%% === Controllability check ===
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The system is NOT
controllable.", r_Wc);
end

%%  === Define the structured controllability basis matrix and
transformation matrix ===
```

```
C_matrix = [B(:,1), A*B(:,1), A^2*B(:,1), B(:,2), A*B(:,2),
A^2*B(:,2)]; % Select 6 linearly-independent column vectors to form a
new C square matrix
r_Cm = rank(C_matrix); % Check if C_matrix is full rank
if r_Cm == 6
    disp("The structured controllability basis matrix is legal.");
else
    fprintf("The rank of structured controllability basis matrix is %d.
It's NOT legal.", r_Cm);
end

C_inv = inv(C_matrix);
d_1 = 3; % The number of vectors associated with u_c
d_2 = 3; % The number of vectors associated with u_h
q_1 = C_inv(d_1, :);
q_2 = C_inv(d_1 + d_2, :); % q_1 and q_2 are rows taken from C_inv to
form T
T = [q_1; q_1*A; q_1*A^2; q_2; q_2*A; q_2*A^2;]; % Transformation
matrix

%% === Solve for the state feedback gain K via symbolic pole placement
in controllable canonical form ===
% Construct the characteristic polynomial of the subsystem
poly_1 = (s - desired_poles(1))*(s - desired_poles(2))*(s -
desired_poles(3)); % 3 poles for the first subsystem
poly_2 = (s - desired_poles(4))*(s - desired_poles(5))*(s -
desired_poles(6)); % 3 poles for the second subsystem
coef_1 = double(coeffs(poly_1)); % Get the coefficient [a3,a2,a1,1]
coef_2 = double(coeffs(poly_2)); % Get the coefficient [a6,a5,a4,1]

% Transform the system from its original coordinates to the
controllable canonical form coordinates
A_bar = T*A/T;
B_bar = T*B;

% Controllable canonical form gain matrix K_bar
K_bar=[k_11_bar, k_12_bar, k_13_bar, k_14_bar, k_15_bar, k_16_bar; %
Affect u_c
       k_21_bar, k_22_bar, k_23_bar, k_24_bar, k_25_bar, k_26_bar]; %
Affect u_h
A_d = A_bar - B_bar*K_bar; % Controllable canonical form state matrix
A_d
```

```
% Match the 3rd and 6th rows of A_d to the desired controllable form
target = [A_d(3,:) == -[coef_1(1:3), 0, 0, 0], A_d(6,:) == -[0, 0, 0,
coef_2(1:3)]];
vars = [k_11_bar k_12_bar k_13_bar k_14_bar k_15_bar k_16_bar k_21_bar
k_22_bar k_23_bar k_24_bar k_25_bar k_26_bar];
sol = solve(target, vars);
K_bar_sol = double([sol.k_11_bar sol.k_12_bar sol.k_13_bar sol.k_14_bar
sol.k_15_bar sol.k_16_bar;
                    sol.k_21_bar sol.k_22_bar sol.k_23_bar sol.k_24_bar
sol.k_25_bar sol.k_26_bar]);

K = K_bar_sol*T; % The original gain matrix K

%% === Step response (zero initial condition, step input) ===
A_cl = A - B*K; % The orignal closed-loop matrix A_cl
sys = ss(A_cl, B, C, 0); % Construct a state-space system

t = 0:0.01:10; % Time span
x0 = zeros(6,1); % Initial state set to zero for step response
u_0 = zeros(length(t), 2); % Initial input set to zero for free
response
u_1 = [ones(length(t),1), zeros(length(t),1)]; % r1=[1,0] step function
u_2 = [zeros(length(t),1), ones(length(t),1)]; % r2=[0,1] step function

% Simulate the reponse of the system to u_1 and u_2
[y1, t_out1] = lsim(sys, u_1, t, x0);
[y2, t_out2] = lsim(sys, u_2, t, x0);
figure; plot(t_out1, y1, 'LineWidth', 1.2); title('Step Response: [1 0]
Input'); xlabel('Time (s)'); ylabel('Output y'); grid on;
legend('d(t)', '\phi(t)', '\psi(t)');
figure; plot(t_out2, y2, 'LineWidth', 1.2); title('Step Response: [0 1]
Input'); xlabel('Time (s)'); ylabel('Output y'); grid on;
legend('d(t)', '\phi(t)', '\psi(t)');

%% === Free response (non-zero initial condition, zero input) ===
% Simulate the free reponse of the system with initial state as x_0
[y_free, t_free, x_free] = lsim(sys, u_0, t, x_0);
u_free = -K * x_free';
u_free = u_free';

figure;
plot(t_free, x_free, 'LineWidth', 1.3);
xlabel('Time (s)');
```

```
ylabel('States');
title('Free Response with x_0 ≠ 0 and u = 0');
legend('x_1 = d(t)', 'x_2 = ϕ(t)', 'x_3 = ψ(t)', 'x_4 = d''(t)', 'x_5 =
ϕ''(t)', 'x_6 = ψ''(t)');
grid on;

figure;
plot(t_free, y_free, 'LineWidth', 1.3);
xlabel('Time (s)');
ylabel('Output y = Cx');
title('Output Response with x_0 ≠ 0 and u = 0');
legend('y_1 = d(t)', 'y_2 = ϕ(t)', 'y_3 = ψ(t)');
grid on;

figure;
plot(t_free, u_free, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Control Input u(t)');
title('Control Inputs during Free Response');
legend('u_c(t)', 'u_h(t)');
grid on;

fprintf('Max control input during free response:\n');
fprintf('u_c: %.2f\n', max(abs(u_free(:,1))));
fprintf('u_h: %.2f\n', max(abs(u_free(:,2))));

%% === Design specifications check ===
evaluate_response(y1, t_out1, '[1, 0]');
evaluate_response(y2, t_out2, '[0, 1]');

function evaluate_response(y, t, label)
    fprintf('\n=== Performance Report for Input %s ===\n', label);
    state_names = {'d(t)', 'phi(t)', 'psi(t)'};

    for i = 1:3
        s = y(:, i);
        final = mean(s(end - floor(length(s)*0.1):end)); % Use average
value to calculate the steady-state value, increase robustness
        peak = max(abs(s)); % Calculate the maximum offset value
(positive/negative)
        overshoot = (peak - abs(final)) / abs(final) * 100; % Calculate
the overshoot
```

```
        band = 0.02 * abs(final); % 2% interval
        settled_idx = find(abs(s - final) <= band); % Get the indices of
those who fall into +-2% interval
        if isempty(settled_idx)
            T_s = NaN; % Not converge
        else
            d_idx = find(diff(settled_idx) > 1, 1, 'last'); % Find the
moment when the system is finally stable
            if isempty(d_idx)
                T_s = t(settled_idx(1));
            else
                T_s = t(settled_idx(d_idx + 1));
            end
        end

        fprintf('Output: %-6s | Overshoot: %7.2f%% | Settling
Time: %5.2fs\n', ...
            state_names{i}, overshoot, T_s);
    end
end
```

task2.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
x_0 = output{5};

%% === Controllability check ===
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The system is NOT
controllable.", r_Wc);
end
```

```
%% === LQR design and solve for the state feedback gain K ===
Q = diag([30, 20, 5, 10, 10, 10]); % State weighting matrix Q: penalize
deviations in systems states
R = diag([5, 5]); % Control input weighting matrix R: penalize control
energy

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma = [A, -B/R*B';
        -Q, -A'];

[eigen_vector, eigen_value] = eig(Gamma);

% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx = find(real(diag(eigen_value))<0);
vector_stable = eigen_vector(:, stable_idx);

n_state = size(A, 1);
v_vector = vector_stable(1:n_state, :); % Upper block v
mu_vector = vector_stable(n_state + 1:end, :); % Lower block mu
P = real(mu_vector/v_vector);
K = R\B'*P; % Get the optimal LQR feedback gain matrix K

%% === Step response (zero initial condition, step input) ===
A_cl = A - B * K; % The orignal closed-loop matrix A_cl
sys = ss(A_cl, B, C, 0); % Construct a state-space system

t = 0:0.01:10; % Time span
x0 = zeros(6,1); % Initial state set to zero for step response
u_0 = zeros(length(t), 2); % Initial input set to zero for free
response
u_1 = [ones(length(t),1), zeros(length(t),1)]; % r1=[1,0] step function
u_2 = [zeros(length(t),1), ones(length(t),1)]; % r2=[0,1] step function

% Simulate the reponse of the system to u_1 and u_2
[y1, t_out1] = lsim(sys, u_1, t, x0);
[y2, t_out2] = lsim(sys, u_2, t, x0);
figure; plot(t_out1, y1, 'LineWidth', 1.2); title('Step Response: [1 0]
Input'); xlabel('Time (s)'); ylabel('Output y'); grid on;
legend('d(t)', '\phi(t)', '\psi(t)');
```

```
figure; plot(t_out2, y2, 'LineWidth', 1.2); title('Step Response: [0 1]
Input'); xlabel('Time (s)'); ylabel('Output y'); grid on;
legend('d(t)', '\phi(t)', '\psi(t)');

%% === Free response (non-zero initial condition, zero input) ===
% Simulate the free reponse of the system with initial state as x_0
[y_free, t_free, x_free] = lsim(sys, u_0, t, x_0);
u_free = -K * x_free';
u_free = u_free';

figure;
plot(t_free, x_free, 'LineWidth', 1.3);
xlabel('Time (s)');
ylabel('States');
title('Free Response with x_0 ≠ 0 and u = 0');
legend('x_1 = d(t)', 'x_2 = ϕ(t)', 'x_3 = ψ(t)', 'x_4 = d''(t)', 'x_5 =
ϕ''(t)', 'x_6 = ψ''(t)');
grid on;

figure;
plot(t_free, y_free, 'LineWidth', 1.3);
xlabel('Time (s)');
ylabel('Output y = Cx');
title('Output Response with x_0 ≠ 0 and u = 0');
legend('y_1 = d(t)', 'y_2 = ϕ(t)', 'y_3 = ψ(t)');
grid on;

figure;
plot(t_free, u_free, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Control Input u(t)');
title('Control Inputs during Free Response');
legend('u_c(t)', 'u_h(t)');
grid on;

fprintf('Max control input during free response:\n');
fprintf('u_c: %.2f\n', max(abs(u_free(:,1))));
fprintf('u_h: %.2f\n', max(abs(u_free(:,2))));

%% === Design specifications check ===
evaluate_response(y1, t_out1, '[1, 0]');
evaluate_response(y2, t_out2, '[0, 1]');
```

```
function evaluate_response(y, t, label)
    fprintf('\n=== Performance Report for Input %s ===\n', label);
    state_names = {'d(t)', 'phi(t)', 'psi(t)'};

    for i = 1:3
        s = y(:, i);
        final = mean(s(end - floor(length(s)*0.1):end)); % Use average
value to calculate the steady-state value, increase robustness
        peak = max(abs(s)); % Calculate the maximum offset value
(positive/negative)
        overshoot = (peak - abs(final)) / abs(final) * 100; % Calculate
the overshoot

        band = 0.02 * abs(final); % 2% interval
        settled_idx = find(abs(s - final) <= band); % Get the indices of
those who fall into +-2% interval
        if isempty(settled_idx)
            T_s = NaN; % Not converge
        else
            d_idx = find(diff(settled_idx) > 1, 1, 'last'); % Find the
moment when the system is finally stable
            if isempty(d_idx)
                T_s = t(settled_idx(1));
            else
                T_s = t(settled_idx(d_idx + 1));
            end
        end

        fprintf('Output: %-6s | Overshoot: %7.2f%% | Settling
Time: %5.2fs\n', ...
            state_names{i}, overshoot, T_s);
    end
end
```

task3.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
syms s;
```

```
syms k_tilde_11_bar k_tilde_12_bar k_tilde_13_bar k_tilde_14_bar
k_tilde_15_bar k_tilde_16_bar k_tilde_21_bar k_tilde_22_bar
k_tilde_23_bar k_tilde_24_bar k_tilde_25_bar k_tilde_26_bar
k_tilde_31_bar k_tilde_32_bar k_tilde_33_bar k_tilde_34_bar
k_tilde_35_bar k_tilde_36_bar;
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
x_0 = output{5};

%% === Controllability check ===
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The system is NOT
controllable.", r_Wc);
end

%% === Observability check ===
W_o = [C; C*A; C*A^2; C*A^3; C*A^4; C*A^5]; % Observability matrix
r_Wo = rank(W_o); % Check if obervability matrix is full rank
if r_Wo == 6
    disp("The system is observable.");
else
    fprintf("The rank of observability matrix is %d. The system is NOT
observable.", r_Wo);
end

%% === LQR design and solve for the state feedback gain K ===
Q = diag([30, 20, 5, 10, 10, 10]); % State weighting matrix Q: penalize
deviations in systems states
R = diag([5, 5]); % Control input weighting matrix R: penalize control
energy

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma = [A, -B/R*B';
        -Q, -A'];

[eigen_vector, eigen_value] = eig(Gamma);
```

```
% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx = find(real(diag(eigen_value))<0);
vector_stable = eigen_vector(:, stable_idx);

n_state = size(A, 1);
v_vector = vector_stable(1:n_state, :); % Upper block v
mu_vector = vector_stable(n_state + 1:end, :); % Lower block mu
P = real(mu_vector/v_vector);
K = R\B'*P; % Get the optimal LQR feedback gain matrix K

%% === Controllability check ===
A_tilde = A';
B_tilde = C';

W_est_c = [B_tilde, A_tilde*B_tilde, A_tilde^2*B_tilde,
A_tilde^3*B_tilde, A_tilde^4*B_tilde, A_tilde^5*B_tilde]; %
Controllability matrix
r_W_est_c = rank(W_est_c); % Check if controllability matrix is full
rank
if r_W_est_c == 6
    disp("The dual system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The dual system
is NOT controllable.", r_W_est_c);
end

%%  === Define the structured controllability basis matrix and
transformation matrix ===
C_matrix = [B_tilde(:,1), A_tilde*B_tilde(:,1), B_tilde(:,2),
A_tilde*B_tilde(:,2), B_tilde(:,3), A_tilde*B_tilde(:,3)]; % Select 6
linearly-independent column vectors to form a new C square matrix
r_Cm = rank(C_matrix); % Check if C_matrix is full rank
if r_Cm == 6
    disp("The structured controllability basis matrix is legal.");
else
    fprintf("The rank of structured controllability basis matrix is %d.
It's NOT legal.", r_Cm);
end

C_inv = inv(C_matrix);
d_1 = 2;
```

```
d_2 = 2;
d_3 = 2; % The number of vectors associated with the 3 second-order
subsystems
q_1 = C_inv(d_1, :);
q_2 = C_inv(d_1 + d_2, :);
q_3 = C_inv(d_1 + d_2 + d_3, :); % q_1, q_2 and q_3 are rows taken from
C_inv to form T
T = [q_1; q_1*A_tilde; q_2; q_2*A_tilde; q_3; q_3*A_tilde]; %
Transformation matrix

%% === Solve for the state feedback gain L via symbolic pole placement
in controllable canonical form ===
desired_poles = [-10, -10, -20, -20, -30, -30];
poly_1 = (s - desired_poles(1))*(s - desired_poles(2)); % 2 poles for
the first subsystem
poly_2 = (s - desired_poles(3))*(s - desired_poles(4)); % 2 poles for
the second subsystem
poly_3 = (s - desired_poles(5))*(s - desired_poles(6)); % 2 poles for
the third subsystem
coef_1 = double(coeffs(poly_1)); % Get the coefficient [a2,a1,1]
coef_2 = double(coeffs(poly_2)); % Get the coefficient [a4,a3,1]
coef_3 = double(coeffs(poly_3)); % Get the coefficient [a6,a5,1]

% Transform the system from its original coordinates to the
controllable canonical form coordinates
A_tilde_bar = T*A_tilde/T;
B_tilde_bar = T*B_tilde;

% Controllable canonical form gain matrix K_tilde_bar
K_tilde_bar = [k_tilde_11_bar, k_tilde_12_bar, k_tilde_13_bar,
k_tilde_14_bar, k_tilde_15_bar, k_tilde_16_bar;
               k_tilde_21_bar, k_tilde_22_bar, k_tilde_23_bar,
k_tilde_24_bar, k_tilde_25_bar, k_tilde_26_bar;
               k_tilde_31_bar, k_tilde_32_bar, k_tilde_33_bar,
k_tilde_34_bar, k_tilde_35_bar, k_tilde_36_bar];

A_tilde_d = A_tilde_bar - B_tilde_bar*K_tilde_bar; % Controllable
canonical form state matrix A_tilde_d

target = [ A_tilde_d(2,:) == -[coef_1(1:2),0, 0, 0, 0], A_tilde_d(4,:)
== -[0, 0, coef_2(1:2), 0, 0], A_tilde_d(6,:) == -[0, 0, 0, 0,
coef_3(1:2)] ];
```

```
vars = [k_tilde_11_bar k_tilde_12_bar k_tilde_13_bar k_tilde_14_bar
k_tilde_15_bar k_tilde_16_bar k_tilde_21_bar k_tilde_22_bar
k_tilde_23_bar k_tilde_24_bar k_tilde_25_bar k_tilde_26_bar
k_tilde_31_bar k_tilde_32_bar k_tilde_33_bar k_tilde_34_bar
k_tilde_35_bar k_tilde_36_bar];
sol = solve(target, vars);
K_tilde_bar_sol = double([sol.k_tilde_11_bar sol.k_tilde_12_bar
sol.k_tilde_13_bar sol.k_tilde_14_bar sol.k_tilde_15_bar
sol.k_tilde_16_bar;
                          sol.k_tilde_21_bar sol.k_tilde_22_bar
sol.k_tilde_23_bar sol.k_tilde_24_bar sol.k_tilde_25_bar
sol.k_tilde_26_bar;
                          sol.k_tilde_31_bar sol.k_tilde_32_bar
sol.k_tilde_33_bar sol.k_tilde_34_bar sol.k_tilde_35_bar
sol.k_tilde_36_bar;]);

K_tilde = K_tilde_bar_sol*T; % The original gain matrix K_tilde
L = K_tilde'; % Get the observer gain
```

task4.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
syms s;
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
x_0 = output{5};
C_2 = [1, 0, 0, 0, 0, 0;
       0, 0, 1, 0, 0, 0]; % Only 2 ouputs are considered

%% === Controllability check ===
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The original system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The original
system is NOT controllable.", r_Wc);
```

```
end

%% === Find the relative degree ===
degree = zeros(2, 1); % Initialize the relative degree for each output
channel
for k = 1:2 % Find the smallest i that makes C_2(k,:)*A^(i-1)*B not
zero
    degree(k) = find(arrayfun(@(i) norm(C_2(k,:)*A^(i-1)*B, 'fro') > 1e-
10, 1:6), 1);
end

degree_1 = degree(1); % First relative degree sigma_1
degree_2 = degree(2); % Second relative degree sigma_1

B_star = [C_2(1,:)*A^(degree_1-1)*B;C_2(2,:)*A^(degree_2-1)*B]; % B*
for the decoupled system

phi_poles_1 = [-6, -8]; % Pole placement for the first output channel
phi_poles_2 = [-10, -12]; % Pole placement for the second output
channel

I = eye(size(A, 1)); % Identity matrix I
phi_f1 = (A - phi_poles_1(1)*I)*(A - phi_poles_1(2)*I);
phi_f2 = (A - phi_poles_2(1)*I)*(A - phi_poles_2(2)*I); % Stable
characteristic polynomial

C_star = [C_2(1,:)*phi_f1; C_2(2,:)*phi_f2]; % C* for the decoupled
system
F = inv(B_star); % Feedforward compensation matrix F
K = B_star\C_star; % Feedback gain matrix K

%% === Step response (zero initial condition, step input) ===
Af = A - B * K; % Closed-loop matrix Af
Bf = B * F; % Input matrix Bf
Cf = C_2; % Output matrix Cf
sys_decouple = ss(Af, Bf, Cf, 0); % Create state-space system

t = 0:0.01:10; % Time span
u_0 = zeros(length(t), 2); % Initial input set to zero for free
response
u_1 = [ones(length(t),1), zeros(length(t),1)]; % r1=[1,0] step function
u_2 = [zeros(length(t),1), ones(length(t),1)]; % r2=[0,1] step function
```

```
[y1, t_out1, ~] = lsim(sys_decouple, u_1, t);
figure;
plot(t_out1, y1, 'LineWidth', 1.2);
title('Step Response to r = [1; 0]');
xlabel('Time (s)');
ylabel('Output');
legend('d(t)', '\phi(t)');
grid on;

[y2, t_out2, ~] = lsim(sys_decouple, u_2, t);
figure;
plot(t_out2, y2, 'LineWidth', 1.2);
title('Step Response to r = [0; 1]');
xlabel('Time (s)');
ylabel('Output');
legend('d(t)', '\phi(t)');
grid on;

%% === Free response (non-zero initial condition, zero input) ===
% Simulate the free reponse of the system with initial state as x_0
[y_free, t_free, x_free] = lsim(sys_decouple, u_0, t, x_0);
u_free = -K * x_free';
u_free = u_free';

figure;
plot(t_free, x_free, 'LineWidth', 1.2);
title('State Response: x(0) ≠ 0, r = 0');
xlabel('Time (s)');
ylabel('States');
legend('x_1 = d(t)', 'x_2 = ϕ(t)', 'x_3 = ψ(t)', 'x_4 = d''(t)', 'x_5 =
ϕ''(t)', 'x_6 = ψ''(t)', 'Location', 'southwest');
grid on;

figure;
plot(t_free, y_free, 'LineWidth', 1.2);
title('Output Response: x(0) ≠ 0, r = 0');
xlabel('Time (s)');
ylabel('Output');
legend('d(t)', '\phi(t)');
grid on;

figure;
plot(t_free, u_free, 'LineWidth', 1.2);
```

```
xlabel('Time (s)');
ylabel('Control Input u(t)');
title('Control Inputs during Free Response');
legend('u_c(t)', 'u_h(t)');
grid on;
```

task5.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
y_sp = output{4};
x_0 = output{5};
w = [-1;1]; % Disturbance (step disturbance at 10s)

%% === Augmented system design ===
% Preparations for the augamented system
A_bar = [A, zeros(6, 3);
        -C, zeros(3, 3)];
B_bar = [B; zeros(3,2)];
B_wbar = [B; zeros(3,2)];
B_rbar = [zeros(6,3); eye(3)];
C_bar = [C, zeros(3,3)];

%% === Controllability check ===
% Check controllability of original system
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The system is NOT
controllable.", r_Wc);
end

% Check controllability of the augmented system
W_qc = [A, B;
```

```
        C, zeros(3,2)];
r_Wqc = rank(W_qc); % Check if controllability matrix is full rank
if r_Wqc == 8
    disp("The augmented system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The augmented
system is NOT controllable.", r_Wqc);
end

%% === LQR design and solve for the state feedback gain K ===
Q = diag([30, 20, 5, 10, 10, 10, 10, 60, 80]); % State weighting matrix
Q: penalize deviations in systems states
R = diag([5, 5]); % Control input weighting matrix R: penalize control
energy

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma = [A_bar, -B_bar/R*B_bar';
        -Q, -A_bar'];

[eigen_vector, eigen_value] = eig(Gamma);

% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx = find(real(diag(eigen_value))<0);
vector_stable = eigen_vector(:, stable_idx);

n_state = size(A_bar, 1);
v_vector = vector_stable(1:n_state, :); % Upper block v
mu_vector = vector_stable(n_state + 1:end, :); % Lower block mu
P = real(mu_vector/v_vector);
K = R\B_bar'*P; % Get the optimal LQR feedback gain matrix K

%% === LQR design and solve for the state feedback gain L ===
A_tilde = A';
B_tilde = C';
Q_tilde = eye(6);
R_tilde = eye(3);

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma_tilde = [A_tilde, -B_tilde/R_tilde*B_tilde';
        -Q_tilde, -A_tilde'];
```

```
[eigen_vector_tilde, eigen_value_tilde] = eig(Gamma_tilde);

% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx_tilde = find(real(diag(eigen_value_tilde))<0);
vector_stable_tilde = eigen_vector_tilde(:, stable_idx_tilde);

n_state_tilde = size(A, 1);
v_vector_tilde = vector_stable_tilde(1:n_state_tilde, :); % Upper block
v
mu_vector_tilde = vector_stable_tilde(n_state_tilde + 1:end, :); %
Lower block mu
P_tilde = real(mu_vector_tilde/v_vector_tilde);
K_tilde = R_tilde\B_tilde'*P_tilde; % Get the optimal LQR feedback gain
matrix K_tilde
L = K_tilde';
```

task6.m

```
clc;
clear;
close all;

%% === Load the predefined parameters ===
output = parameter_init();
A = output{1};
B = output{2};
C = output{3};
x_0 = output{5};
w = [-1;1]; % Disturbance (step disturbance at 10s)
y_sp = [1;2;3];
%% === Augmented system design ===
% Preparations for the augamented system
A_bar = [A, zeros(6, 3);
        -C, zeros(3, 3)];
B_bar = [B; zeros(3,2)];
B_wbar = [B; zeros(3,2)];
B_rbar = [zeros(6,3); eye(3)];
C_bar = [C, zeros(3,3)];

%% === Controllability check ===
% Check controllability of original system
```

```
W_c = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B]; % Controllability matrix
r_Wc = rank(W_c); % Check if controllability matrix is full rank
if r_Wc == 6
    disp("The system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The system is NOT
controllable.", r_Wc);
end

% Check controllability of the augmented system
W_qc = [A, B;
        C, zeros(3,2)];
r_Wqc = rank(W_qc); % Check if controllability matrix is full rank
if r_Wqc == 8
    disp("The augmented system is controllable.");
else
    fprintf("The rank of controllability matrix is %d. The augmented
system is NOT controllable.", r_Wqc);
end

%% === LQR design and solve for the state feedback gain K ===
Q = diag([30, 20, 5, 10, 10, 10, 10, 60, 80]); % State weighting matrix
Q: penalize deviations in systems states
R = diag([5, 5]); % Control input weighting matrix R: penalize control
energy

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma = [A_bar, -B_bar/R*B_bar';
        -Q, -A_bar'];

[eigen_vector, eigen_value] = eig(Gamma);

% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx = find(real(diag(eigen_value))<0);
vector_stable = eigen_vector(:, stable_idx);

n_state = size(A_bar, 1);
v_vector = vector_stable(1:n_state, :); % Upper block v
mu_vector = vector_stable(n_state + 1:end, :); % Lower block mu
P = real(mu_vector/v_vector);
K = R\B_bar'*P; % Get the optimal LQR feedback gain matrix K
```

```
%% === LQR design and solve for the state feedback gain L ===
A_tilde = A';
B_tilde = C';
Q_tilde = eye(6);
R_tilde = eye(3);

% Generalized Hamiltonian matrix Gamma used for solving the Algebraic
Riccati Equation
Gamma_tilde = [A_tilde, -B_tilde/R_tilde*B_tilde';
        -Q_tilde, -A_tilde'];

[eigen_vector_tilde, eigen_value_tilde] = eig(Gamma_tilde);

% Find the eigenvectors associated with eigenvalues that have negative
real parts
stable_idx_tilde = find(real(diag(eigen_value_tilde))<0);
vector_stable_tilde = eigen_vector_tilde(:, stable_idx_tilde);

n_state_tilde = size(A, 1);
v_vector_tilde = vector_stable_tilde(1:n_state_tilde, :); % Upper block
v
mu_vector_tilde = vector_stable_tilde(n_state_tilde + 1:end, :); %
Lower block mu
P_tilde = real(mu_vector_tilde/v_vector_tilde);
K_tilde = R_tilde\B_tilde'*P_tilde; % Get the optimal LQR feedback gain
matrix K_tilde
L = K_tilde';
```