EE5101 Linear Systems

Mini-project

《Control of a Stationary Self-Balancing Two-wheeled Vehicle》

2022.11

Abstract

Self-sustaining two-wheeled vehicle not only is a proof of how control theory has been developed during the past decades, but also has a huge market potential. In this mini-project, a control system is designed for it. After given a state space model, various kinds of feedback controllers are applied. The report illustrates the result of the project tasks, and discussed the implement after changing the parameters. During the process of completing the tasks, a deeper understand of the linear system is obtained. There are six sections of the project, which are shown as following:

1. A state feedback controller using the pole place method.
2. A state feedback controller using the LQR method.
3. A state observer and the resultant observer based LQR control system.
4. A decoupling controller with closed-loop stability and resultant control system
5. A controller at steady state with plant operating around the set point.
6. The proof maintaining three outputs at an arbitrary constant set point with zero steady-state error is impossible.

Contents

Figure catalogue

Table catalogue

1.Introduction

1.1 Background

Self-sustaining two-wheeled vehicle not only is a proof of how control theory has been developed during the past decades, but also has a huge market potential. Therefore, a lot of researchers from universities and companies are working on related topics. It is a practical attempt to utilize control theory to complete a self-sustaining two-wheeled vehicle.

1.2 Modeling

For model-based control, the first step is to build an effective dynamic model for our target plant, i.e., the two-wheeled vehicle in this project. The detailed procedures to model this vehicle can be found in [1] and [2]. An experimental system for the two-wheeled vehicle prototype is shown in Figure 1-1. The two- wheeled vehicle consists of three parts. There is a cart system that corresponds to the rider's center- of-gravity movement, a steering system (a front part) for steering, and a body (a rear part). The front part and the rear part are structures that are movable through a steering axis. A cart system and a steering system are driven by DC servo motor, and DC motors are controlled by servo amplifier which contains the velocity control system. Handle angle and cart position are measured by encoders. Attitude angles of the two-wheeled vehicle (roll angle and yaw angle) are measured by gyroscopes.

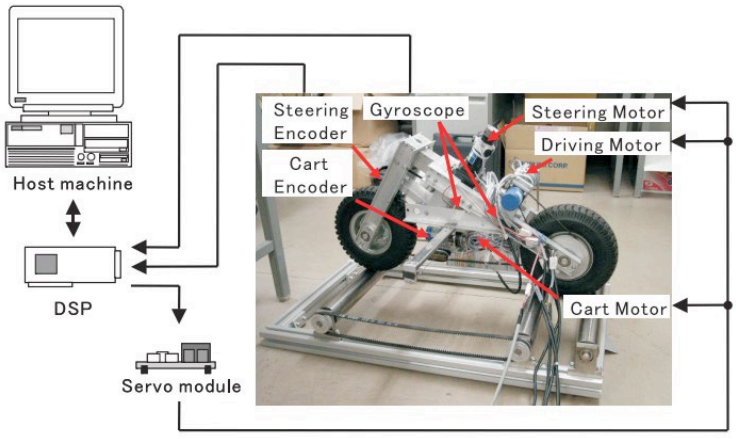

Figure 1-1 Composition of experimental system

The mechanical structure for the two-wheeled vehicle is given in Figure 1-2. T 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.

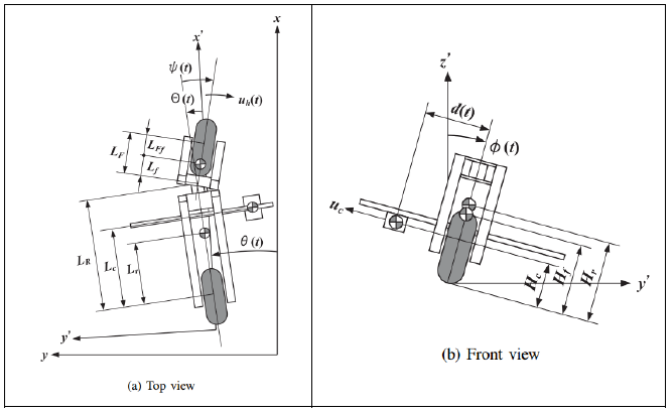

Figure 1-2 Two-wheeled vehicle structure model

In [1], the state space linear model for the two-wheeled vehicle is derived to be

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{0-1}$$

Where the state variable is

$$x = \begin{bmatrix} d(t) & \phi(t) & \psi(t) & \dot{d}(t) & \dot{\phi}(t) & \dot{\psi}(t) \end{bmatrix} \tag{0-2}$$

Since my Student ID is A0260074M, then $a = 0,\ b = 0,\ c = 7,\ d = 4$. So, my matrices and the input vector are:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 6.5 & -10 & -15.5 & 0 & 0 \\ -21.84 & 19.71 & 6.07 & -3.38 & -4.26 & 0.30 \\ 5 & -3.6 & 0 & 0 & 0 & -10.5 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 25.5 & 11.2 \\ 5.57 & -1.72 \\ 40 & 60 \end{bmatrix} \tag{1-3}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, u = \begin{bmatrix} u_c(t) & u_h(t) \end{bmatrix}^T \tag{0-3}$$

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

$$x_0 = \begin{bmatrix} 0.2 & -0.1 & 0.15 & -1 & 0.8 & 0 \end{bmatrix}^T \tag{0-4}$$

2. Pole Placement Method

2.1 State feedback controller design

To obtain proper poles, there are some control specifications needing to be considered. In this project, the transient response performance specifications for all the outputs y in state space model are as follows:

1. The overshoot is less than 10%.

$$M_p = e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}} \leq 0.1 \tag{2-1}$$

2.The 2% settling time is less than 5 seconds

$$t_s = \frac{4.0}{\xi\omega_n} \leq 5 \tag{2-2}$$

Since a standard second-order system is considered as:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2-3}$$

Poles are at:

$$-\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \text{ (with } 0 < \zeta < 1) \tag{2-4}$$

After calculation, the range of ξ and $\xi\omega_n$ should satisfy:

$$\begin{cases} \xi > 0.5912 \\ \xi\omega_n > 0.8 \end{cases} \tag{2-5}$$

And my choice of them is:

$$\begin{cases} \xi = 0.8 \\ \omega_n = 1 \end{cases} \tag{2-6}$$

The poles of the reference model are

$$s = -0.8 \pm 0.6i \tag{2-7}$$

1) For this part, assume that I can measure all the six state variables. For the six-order system, I choose the desired poles are:

TABLE 2-1 Designed pole position

S1	-0.8+0.6i
S2	-0.8-0.6i
S3	-4
S4	-4
S5	-6
S6	-6

The S3 – S6 is at least 3 to 5 times larger.

2) Check whether the plant is controllable. Calculate the rank of controllability matrix W_c

$$W_c = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & Ab_1 & Ab_2 & A^2b_1 & A^2b_2 & A^3b_1 & A^3b_2 \end{bmatrix} \quad (2\text{-}8)$$

```
>> rank(W)

ans =

     6
```

Figure 2-1 The rank of W_c

Since the rank of W_c is 6, it means system is controllable and the transformation matrix T exists. Then I need to build a transform matrix T. For multi-input cases, I take out 2 rows from C^{-1} corresponding to the 2 inputs. Meanwhile, there is 3 vectors in C are associated with u_c and 3 vectors are associated with u_h. So $d_1 = d_2 = 3$. The matrix T is shown as below:

$$T = \begin{bmatrix} q_3^T \\ q_3^T A \\ q_3^T A^2 \\ q_{3+3}^T \\ q_{3+3}^T A \\ q_{3+3}^T A^2 \end{bmatrix} \quad (2\text{-}9)$$

Then the matrix $\bar{A}$ and $\bar{B}$ are easy to obtain:

$$\bar{A} = TAT^{-1}, \bar{B} = TB \quad (2\text{-}10)$$

Design the feedback gain matrix for the controllable canonical form

$$\bar{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \end{bmatrix} \quad (2\text{-}11)$$

Compare $\bar{A} - \bar{B}\bar{K}$ and A_d. Finally, get K

$$K = \bar{K}T = \begin{bmatrix} -7.51 & 6.81 & 2.81 & -4.64 & 3.15 & 1.86 \\ -18.94 & 19.16 & 8.59 & -1.58 & 2.88 & 0.67 \end{bmatrix}$$

A | B | K | K_calculated

2x6 double

	1	2	3	4	5	6
1	-7.5127	6.8189	2.8142	-4.6456	3.1514	1.8667
2	-18.9408	19.1657	8.5922	-1.5839	2.8843	0.6714

Figure 2-2The calculated K in MATLAB

2.2 Simulation

The initial state is x0, we can use Simulink to simulate the system. My simulate module is showed in the following figure 2-3.

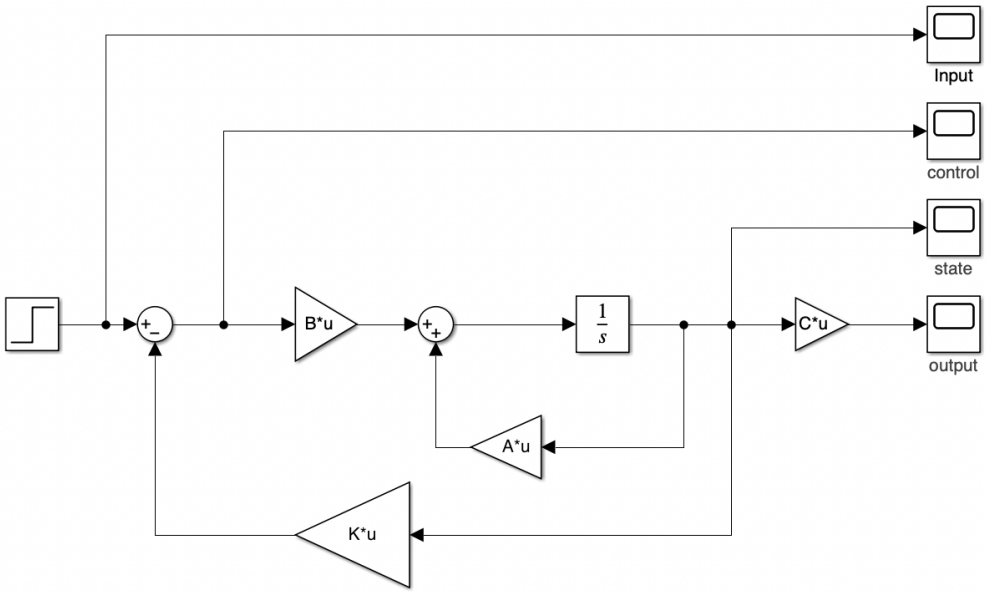

Figure 2-3 Pole placement simulation module

The poles are designed as the table 3-1 shows. The output, control signal and state response of all six which are non-zero initial state with zero external inputs are show as following.

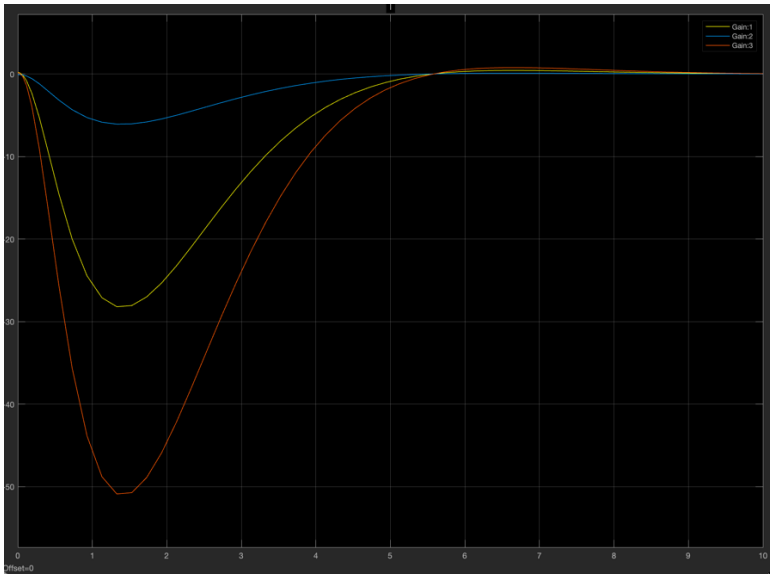

Figure 2-4 Pole placement output with zero input

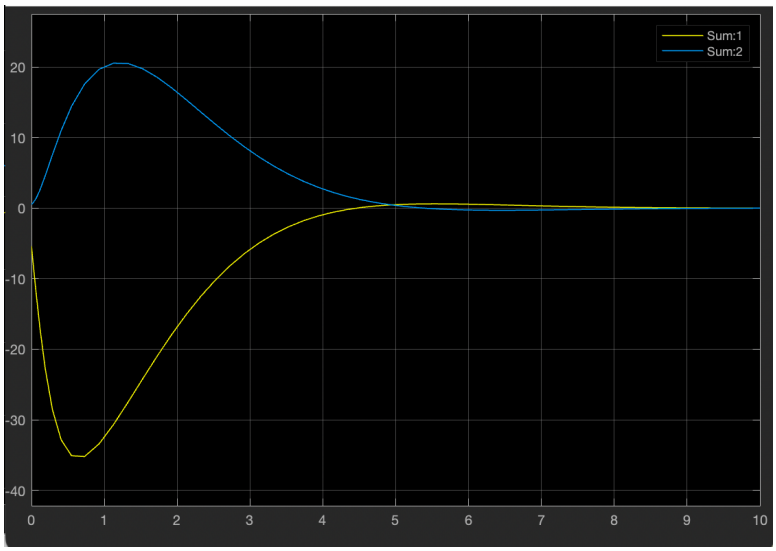

Figure 2-5 Pole placement control signal with zero input

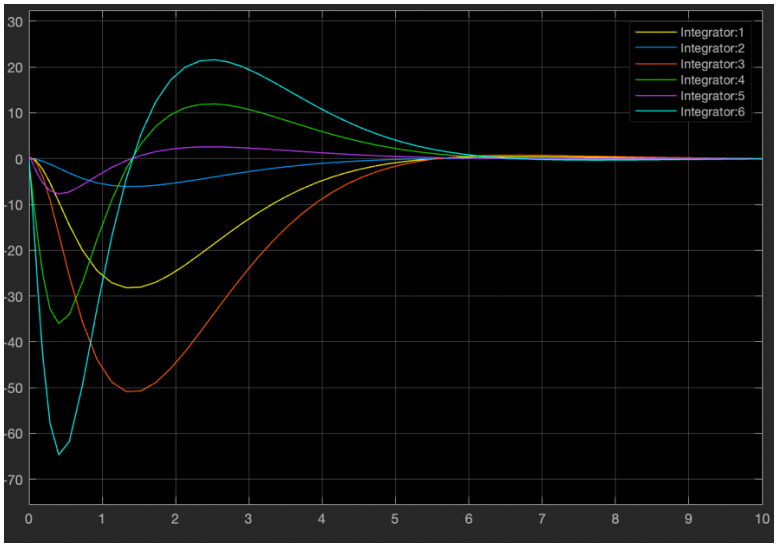

Figure 2-6Pole placement responses with zero input

Then change the input to r = [1,0], and the output of non-zero inputs & zero initial states is shown in figure 2-7.

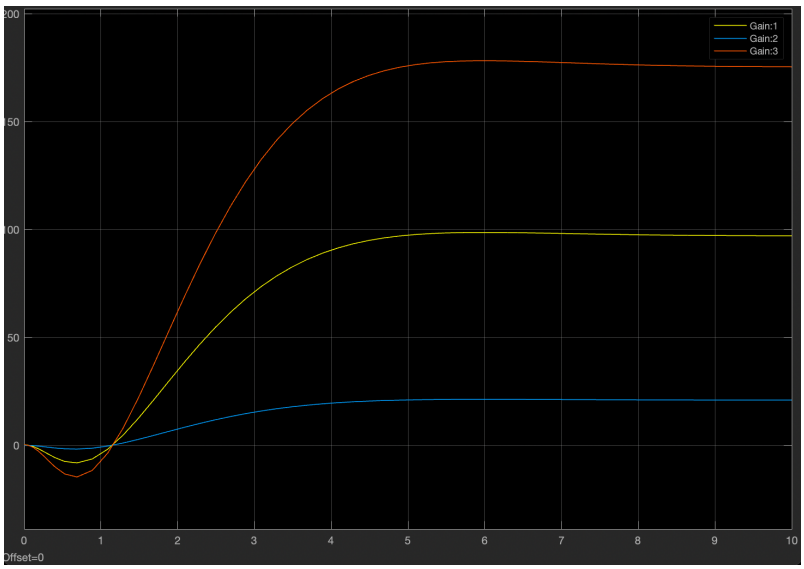

Figure 2-7 Pole placement output with non-zero input and r = [1,0]

Then change the input to r = [0,1], and the output of non-zero inputs & zero initial states is shown in figure 2-8.

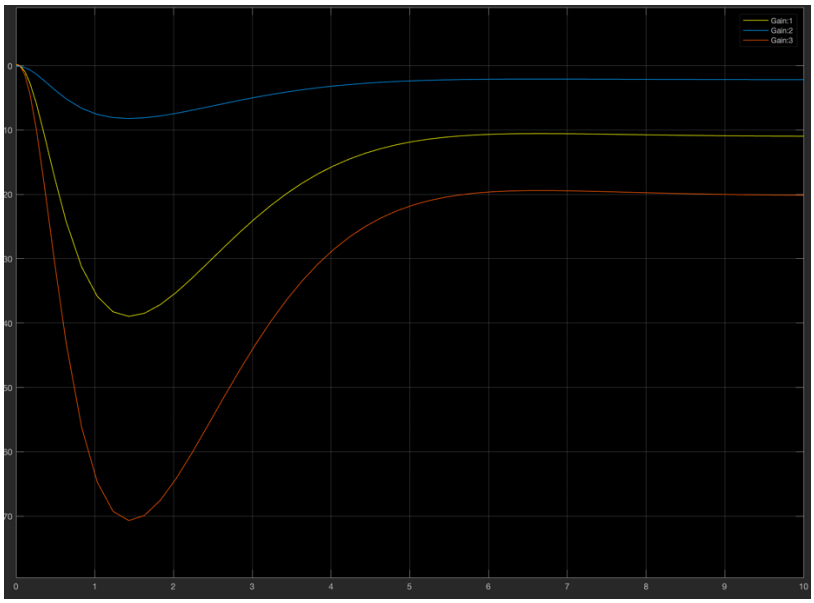

Figure 2-8Pole placement output with non-zero input and r = [0,1]

It is obvious that the overshoot is much larger than 10%. It does not satisfy the requirement. So I change the pole position as table 3-2 shows.

TABLE 2-2 Amended pole placement simulation module

S1	-0.8+0.6i
S2	-0.8-0.6i
S3	-8
S4	-8
S5	-2
S6	-2

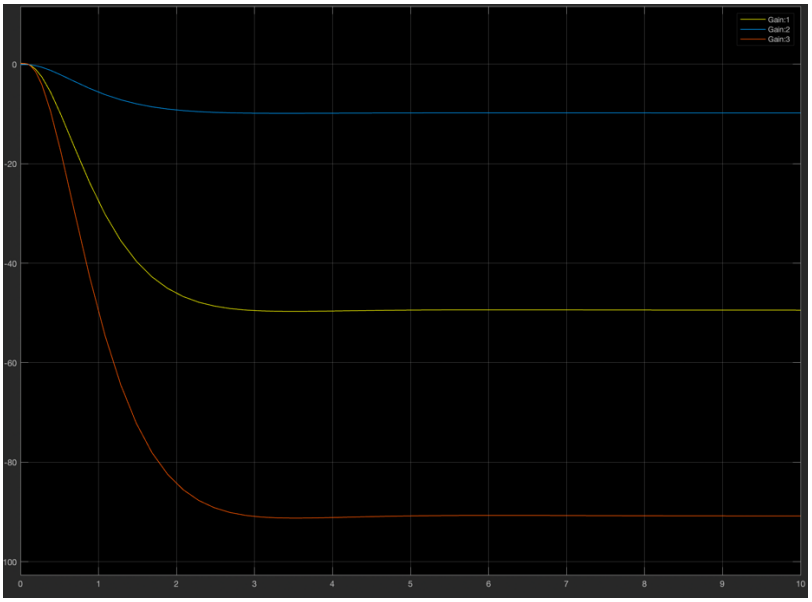

Figure 2-9 Pole placement output with non-zero input and r = [0,1] after changing pole

Now, it performs well. The performance of the new pole placement in zero-input and r =[1,0] are similar to the figure 2-4 and 2-7. To reduce the length of the report, I omit them. The step responses of the three outputs of the new pole placement meet the design specifications. It shows the pole placement is not as larger as better. It should be tested and find a proper value.

2.3 Analysis

Change the pole placement as the table 3-3 shows:

TABLE 2-3 Pole placement: changes of the positions of poles

Poles	s_1	s_2	s_3	s_4	s_5	s_6
Original	$-0.8+0.6i$	$-0.8-0.6i$	-8	-8	-2	-2
Change 1	$-3+0.6i$	$-3-0.6i$				
Change 2	$-0.4+0.6i$	$-0.4-0.6i$				
Change 3	$-0.8+3i$	$-0.8-3i$				
Change 4	$-0.8+0.3i$	$-0.8-0.3i$				

Set the pole placement as change 1:

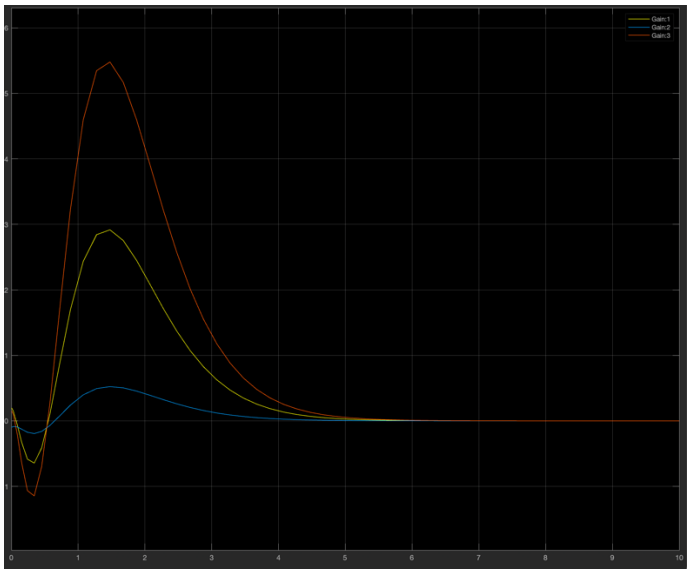

Figure 2-10 Pole placement output with zero input

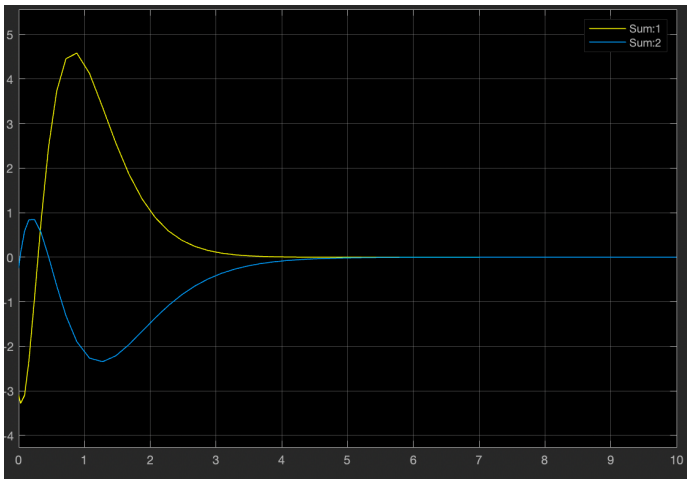

Figure 2-11 Pole placement control signal with zero input

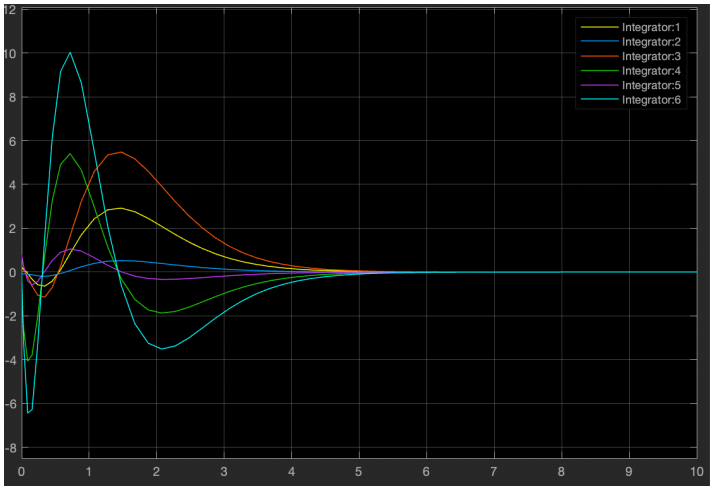

Figure 2-12 Pole placement responses with zero input

Set the pole placement as change 2:

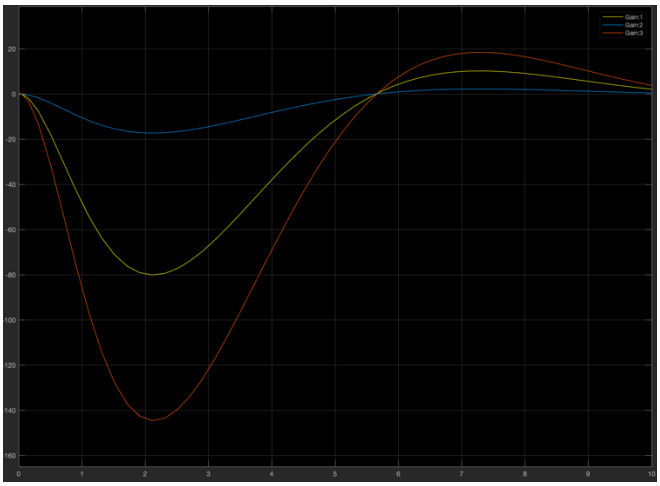

Figure 2-13 Pole placement output with zero input

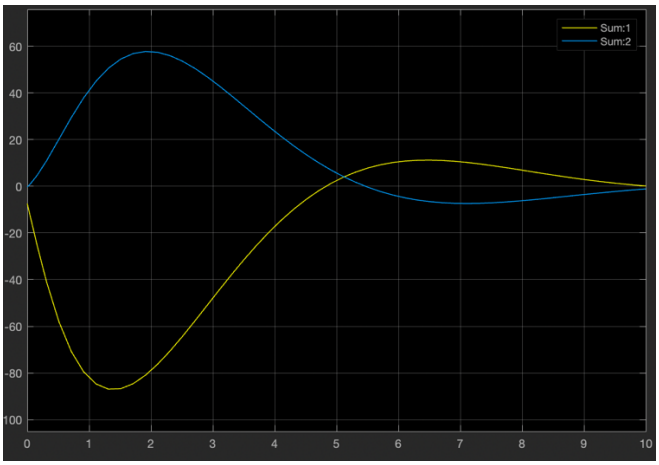

Figure 2-14 Pole placement control signal with zero input

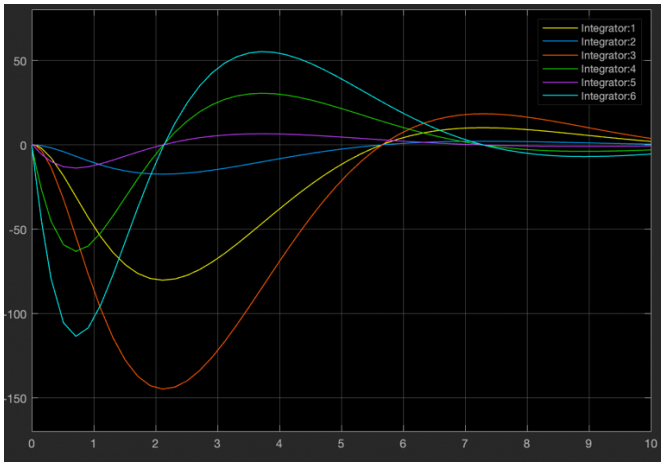

Figure 2-15 Pole placement responses with zero input

Discussion:

If the absolute value of real part of the poles is larger than the point of original pole, the system settling time will be shorter and the input signal will be smaller. If the absolute value of real part of the poles is smaller than the point of original pole, the

system settling time will be longer and the input signal will be larger.

Set the pole placement as change 3:

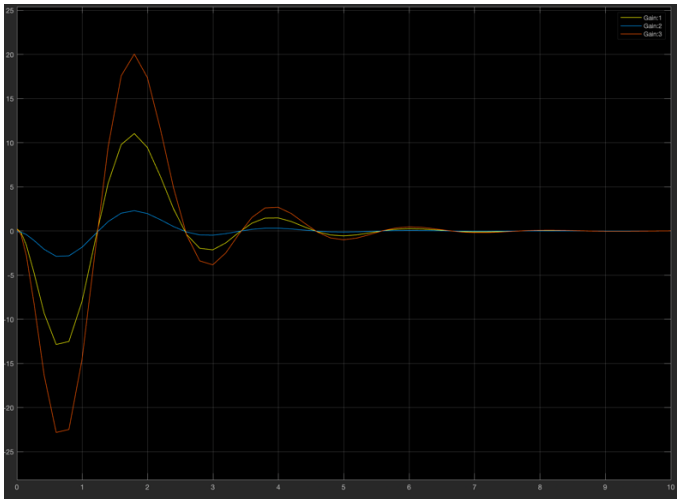

Figure 2-16 Pole placement output with zero input

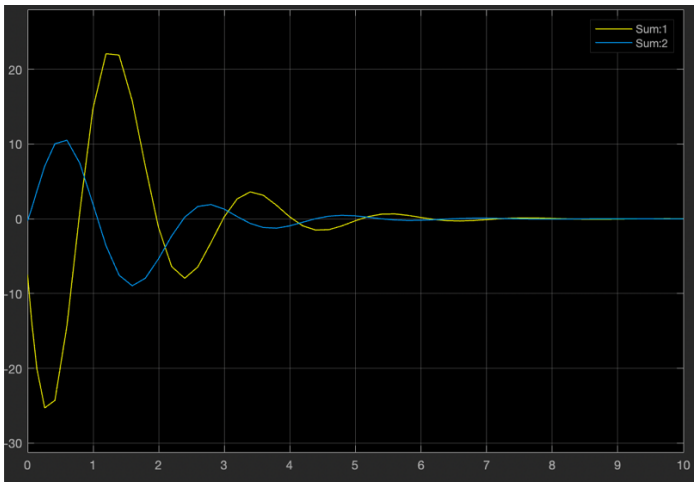

Figure 2-17 Pole placement control signal with zero input

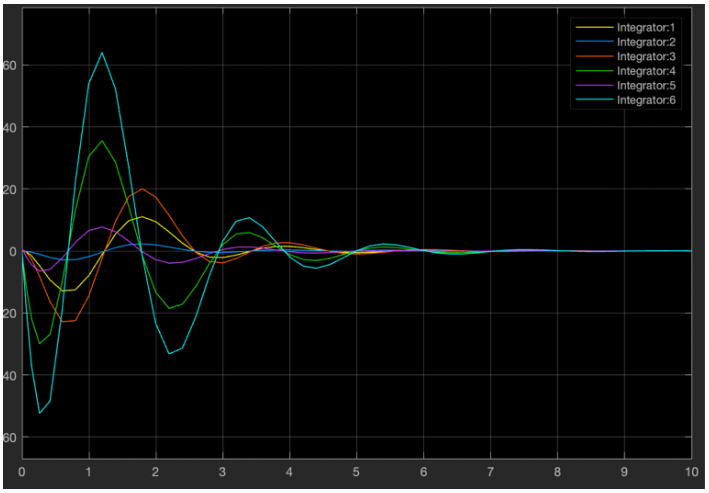

Figure 2-18 Pole placement responses with zero input

Set the pole placement as change 4:

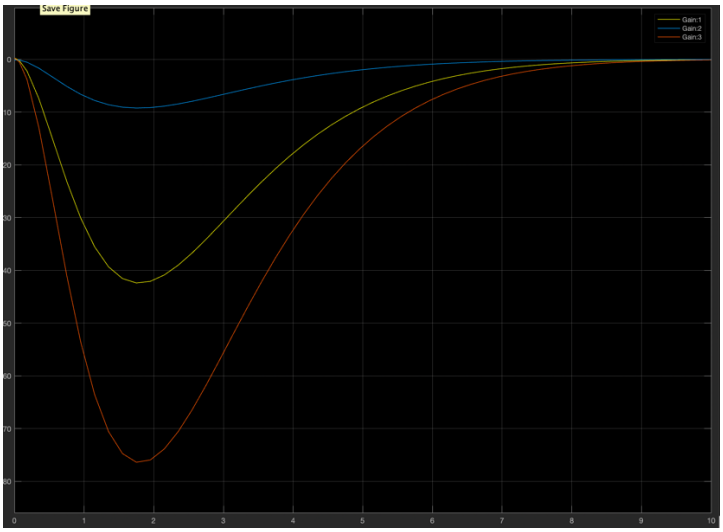

Figure 2-19 Pole placement output with zero input

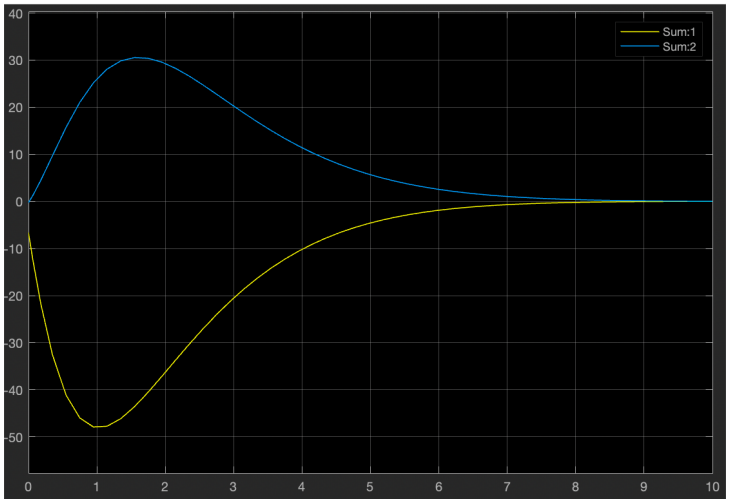

Figure 2-20 Pole placement control signal with zero input

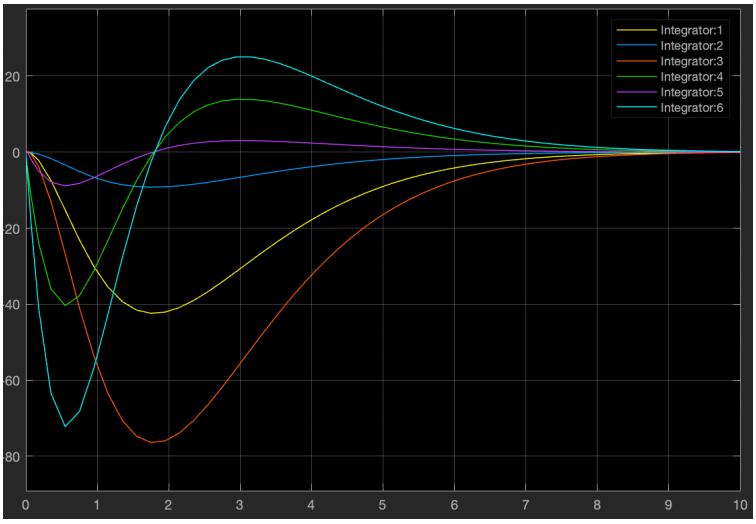

Figure 2-21 Pole placement responses with zero input

Discussion:

If the absolute value of imaginary part of the poles is larger than the point of original pole, the system overshoot will be larger and appear more oscillatory response. If the absolute value of imaginary part of the poles is smaller than the point of original pole, the system overshoot will be smaller and appear less oscillatory response.

3. LQR Method

3.1 State Feedback Controller Design

Assume that I can measure all the six state variables. I need to design a state feedback controller using the LQR method. Assume that the matrix Q and R is:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3-1}$$

The quadratic cost function should be minimized

$$J = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u) dt \tag{3-2}$$

The optimal control law:

$$u = -Kx \tag{3-3}$$

Try the Lyapunov method this time, which is:

$$V(x) = x^T P x \tag{3-4}$$

Where P is a positive matrix and meet the Algebraic Riccati Equation (ARE), which is

$$A^T P + PA + Q = PBR^{-1}B^T P \tag{3-5}$$

To solve the ARE equation, systematic way of eigenvalue-eigenvector based algorithm is usually used. First, define a $2n \times 2n$ matrix Γ

$$\Gamma = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{3-6}$$

Γ is really a big matrix, so I show it in MATLAB.

12x12 double

	1	2	3	4	5	6	7	8	9	10	11	12
2	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0
4	0	6.5000	-10	-15.5000	0	0	0	0	0	-775.6900	-122.7563	-1692
5	-21.8453	19.7139	6.0712	-3.3860	-4.2661	0.3014	0	0	0	-122.7563	-33.9983	-119.4681
6	5	-3.6000	0	0	0	-10.5000	0	0	0	-1692	-119.4681	-5200
7	-1	0	0	0	0	0	0	0	0	0	21.8453	-5
8	0	-10	0	0	0	0	0	0	0	-6.5000	-19.7139	3.6000
9	0	0	-100	0	0	0	0	0	0	10	-6.0712	0
10	0	0	0	-100	0	0	-1	0	0	15.5000	3.3860	0
11	0	0	0	0	-10	0	0	-1	0	0	4.2661	0
12	0	0	0	0	0	-1	0	0	-1	0	-0.3014	10.5000

Figure 3-1 value of matrix Γ

Then let the eigenvector of Γ corresponding to stable $\lambda_i, i = 1,2,3,4,5,6$

12x6 double

	1	2	3	4	5	6
1	-0.0014	-5.2946e...	2.6430e-...	6.3580e-...	0.0020	0.0245
2	-2.2790e...	-0.0022	-0.0106	-0.0297	-0.0012	0.0286
3	-0.0031	0.0244	-0.0716	-4.7473e...	0.0070	0.0017
4	0.4122	0.0215	-0.0027	-0.0043	-0.0049	-0.0093
5	0.0651	0.0908	0.1102	0.2026	0.0030	-0.0109
6	0.8984	-0.9926	0.7412	0.0032	-0.0170	-6.5409e...
7	-1.2478e...	-0.0133	-0.1743	-0.7183	0.7755	0.9984
8	0.0032	0.0169	0.1299	0.5257	-0.6203	0.0311
9	-0.0058	0.0540	-0.6148	0.3239	-0.0669	-0.0169
10	0.1368	0.0369	-0.0284	-0.0865	0.0326	0.0076
11	0.0023	0.0206	0.0843	0.2299	-0.0882	-0.0168
12	0.0030	-0.0182	0.0073	0.0229	-0.0085	-0.0021

$$\begin{pmatrix} v_i \\ \mu_i \end{pmatrix} = \quad (3\text{-}7)$$

P is given by

1	2	3	4	5	6
708.4862	-584.9917	-156.5555	37.9207	-90.7731	-10.3761
-584.9917	513.6313	135.7563	-32.8233	79.0736	8.9964
-156.5555	135.7563	50.2336	-9.5423	21.8278	2.7479
37.9207	-32.8233	-9.5423	2.6209	-5.3055	-0.6465
-90.7731	79.0736	21.8278	-5.3055	12.9067	1.4517
-10.3761	8.9964	2.7479	-0.6465	1.4517	0.1899

$$P = \mu v^{-1} = \quad (3\text{-}8)$$

Finally, the feedback matrix K can be calculated

$$K = -R^{-1}B^T P = \begin{bmatrix} 46.27 & -36.65 & -11.81 & 11.42 & -5.32 & -0.8 \\ -41.49 & 35.95 & 20.4 & 0.29 & 5.44 & 1.65 \end{bmatrix} \quad (3\text{-}9)$$

3.2 Simulation

The initial state is x0, I use Simulink to simulate the system. My simulate module is showed in the following figure 3-2.

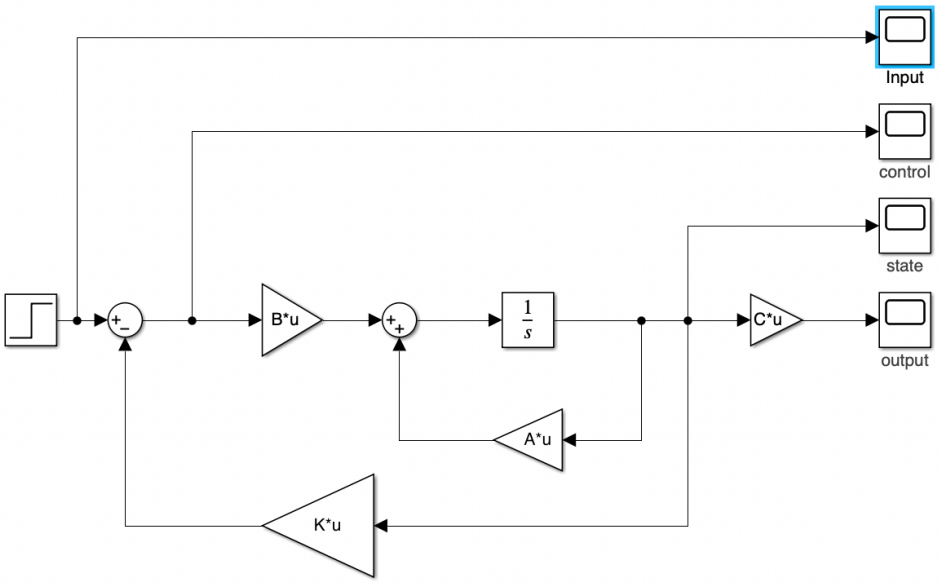

Figure 3-1 LQR simulation module

The Q and R are designed as formula 3-1 shows. The output, control signal and state response of all six which are non-zero initial state with zero external inputs are show

as following.

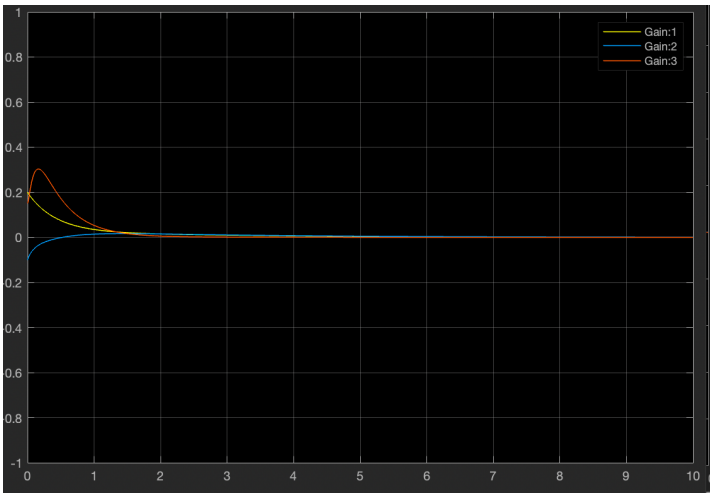

Figure 3-2 LQR output with zero input

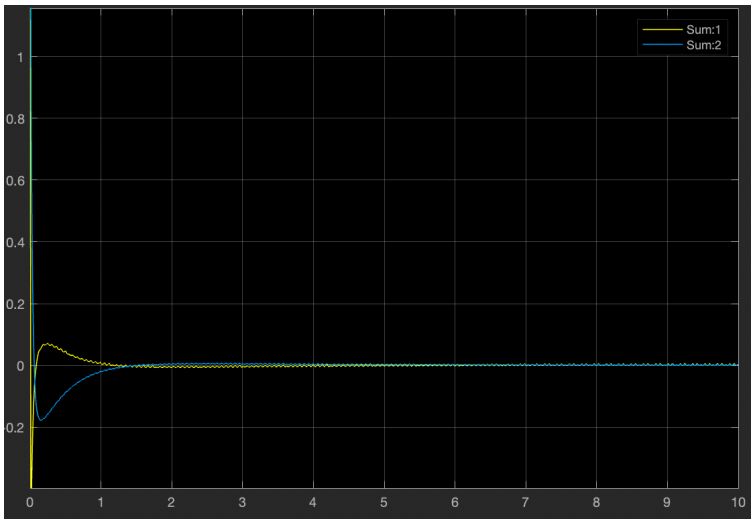

Figure 3-3 LQR control signal with zero input

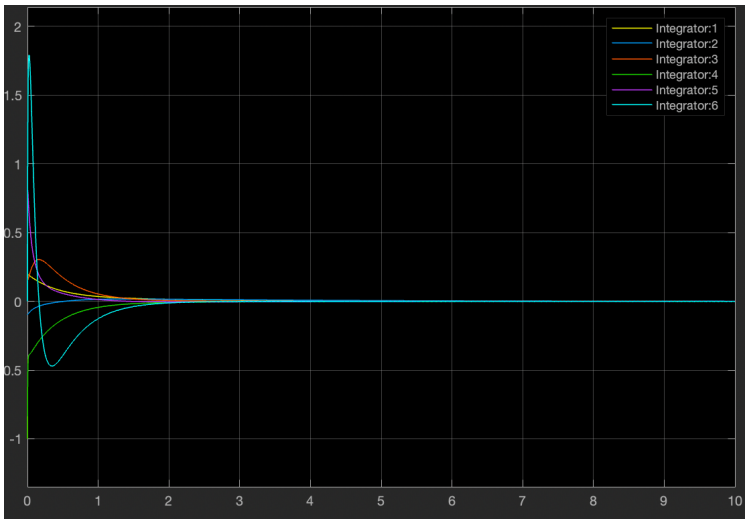

Figure 3-4 LQR responses with zero input

Then change the input to r = [1,0], and the output of non-zero inputs & zero initial states is shown in figure 3-5.

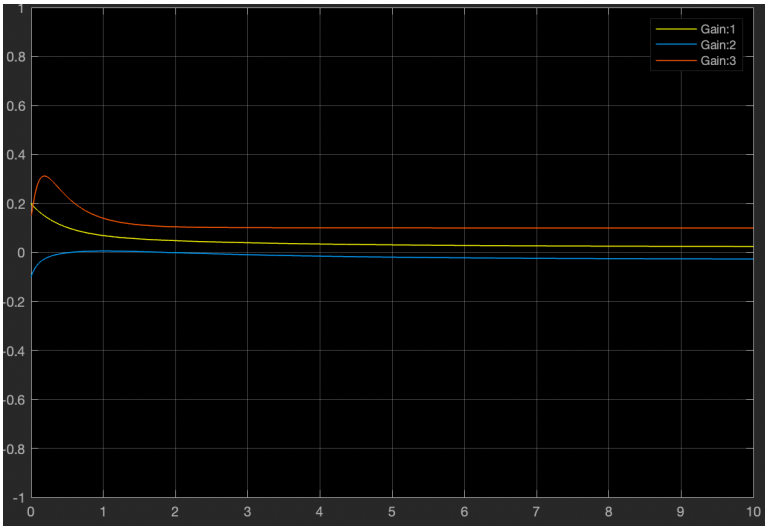

Figure 3-5 LQR responses with non-zero input and r=[1,0]

Then change the input to r = [0,1], and the output of non-zero inputs & zero initial states is shown in figure 3-6.

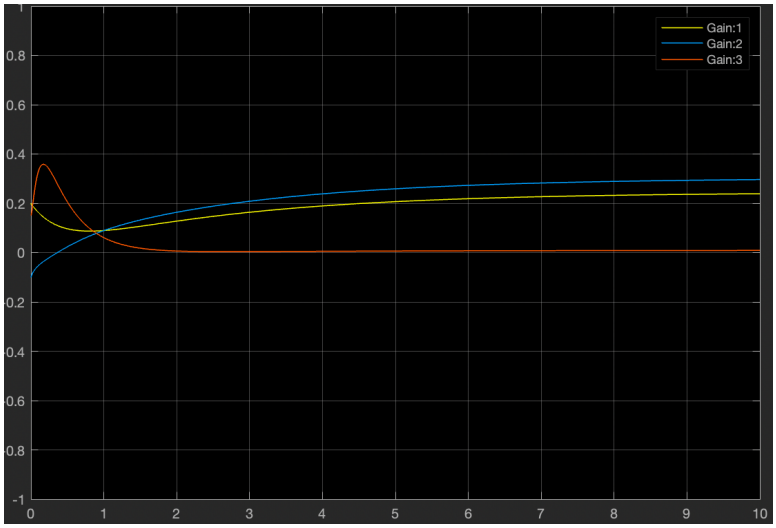

Figure 3-6 LQR responses with non-zero input and r=[0,1]

The step responses of the three outputs of the LQR method designed system meet the design specifications.

3.3 Analysis

3.3.1 Weight of Q

1) Increase the value in Q: (increase q22 from 10 to 100)

The result:

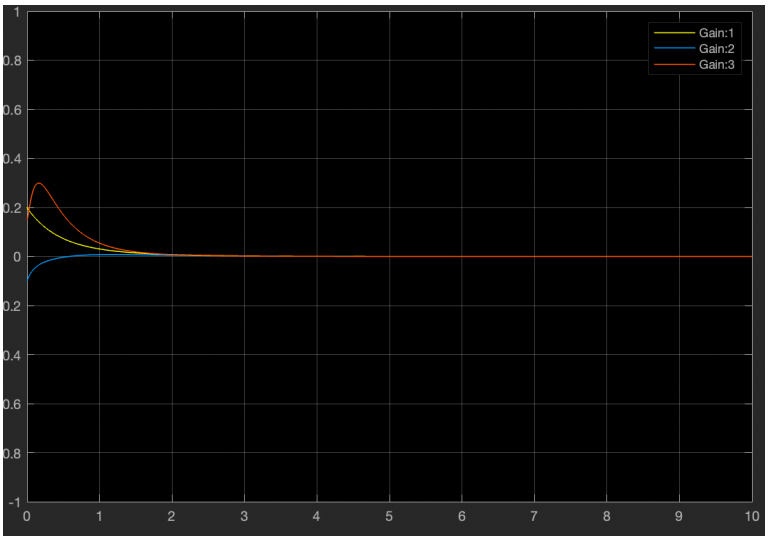

Figure 3-7 LQR output with zero input

2) Decrease the value in Q: (decrease q33 from 100 to 1)

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The result:

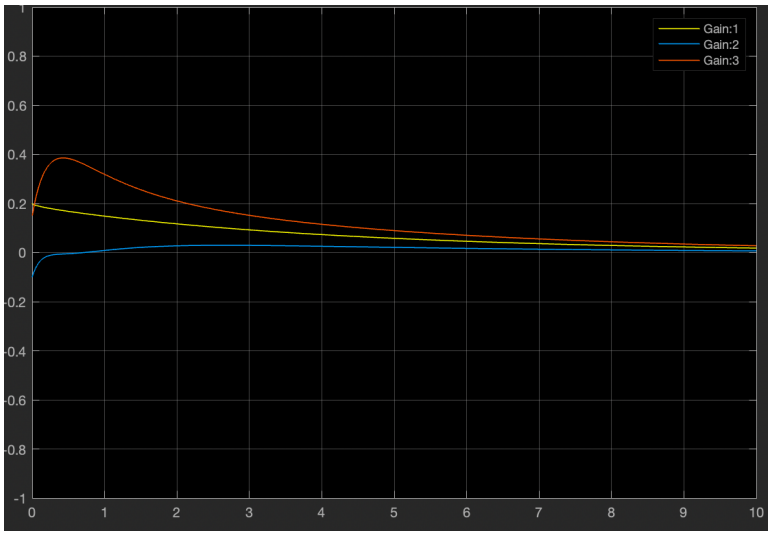

Figure 3-8 LQR output with zero input

Discussion:

If the value increases in Q, the settling time of the system will be shorter and the input signal will be larger. If the value in Q decreases, the settling time of the system will be longer and the input signal will be smaller.

3.3.2 Weight of R

1) Increase the value in R: (r11 increases from 1 to 100)

$$R = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

Input is non-zeros, and the result:

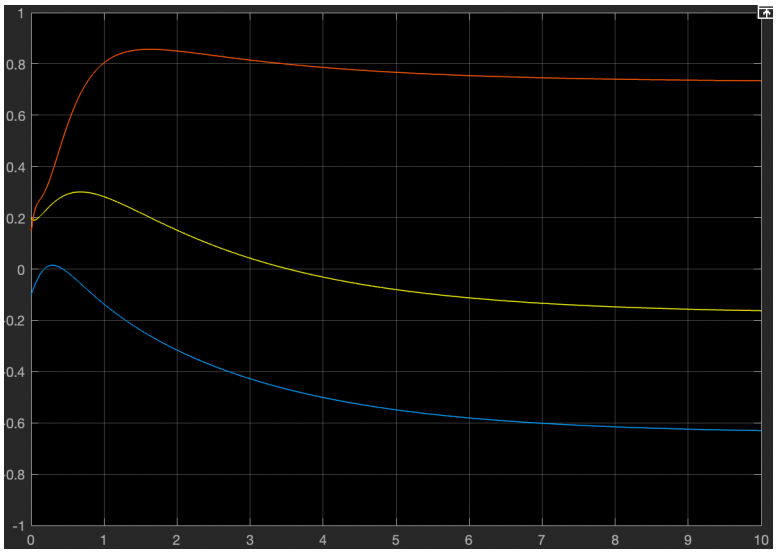

Figure 3-9 LQR output with non-zero input and r =[1,0]

2) Decrease the value in R: (r11 decreases from 1 to 0.01)

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$$

Input is non-zeros, and the result:

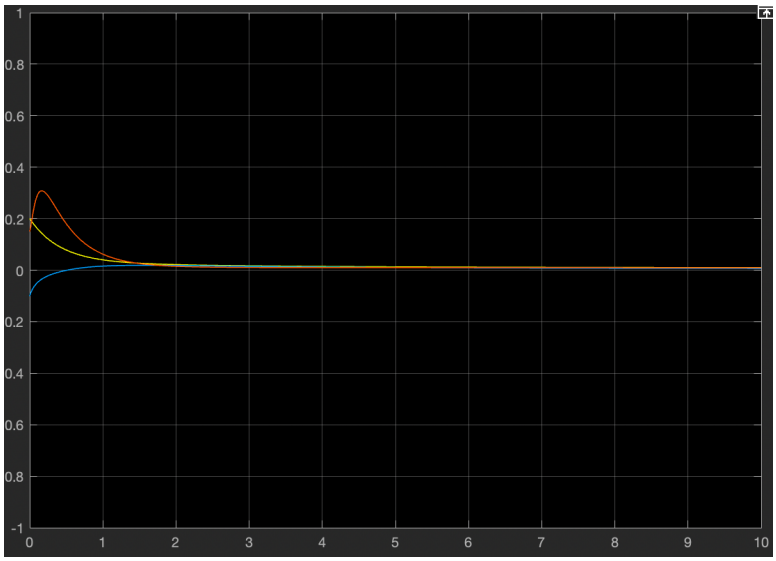

Figure 3-10 LQR output with non-zero input and r =[1,0]

Discussion:

If the value increases in R, the settling time of the system will be longer and the input signal will be smaller. If the value in R decreases, the settling time of the system will be shorter and the input signal will be larger.

Actually, six number in Q matrix corresponds to six states separately. Similarly, two number in matrix corresponds to two input signals separately.

4. State Observer

4.1 Observer Design

In this task, the LQR controller is still used. Since LQR need state feedback, and only measure three of them. We should design an observer to estimate the other states. I utilize pole placement and LQR to design an observer.
The full order controller is designed by LQR, using the following Q, R

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4-1}$$

The full-order observer, also known as a closed-loop estimator, consider an estimator in

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + L[y - \hat{y}] \\ \hat{y} &= C\hat{x} \end{aligned} \tag{4-2}$$

For this task, design

$$\tilde{A} = A^T, \tilde{B} = C^T, \tilde{K} = L^T \tag{4-3}$$

Since the system is controllable, using pole placement method, and set the pole as below:

TABLE 4-1 Designed pole placement

S1	-8
S2	-8
S3	-12
S4	-12
S5	-24
S6	-24

Then I need to build a transform matrix T. For multi-input cases, I take out 3 rows from C^{-1} corresponding to the 3 inputs. So $d_1 = d_2 = d_3 = 2$. And $L = \widetilde{K^T}$:

0.5000	0	0
-3.3860	19.7339	0.3014
0	0	37.5000
56.2500	6.5000	-10
-9.0933	79.5277	16.0896
5	-3.6000	182.2500

Figure 4-1 L value

4.2 Simulation

The initial state is x0, I use Simulink to simulate the system. My simulate module is showed in the following figure 4-2.

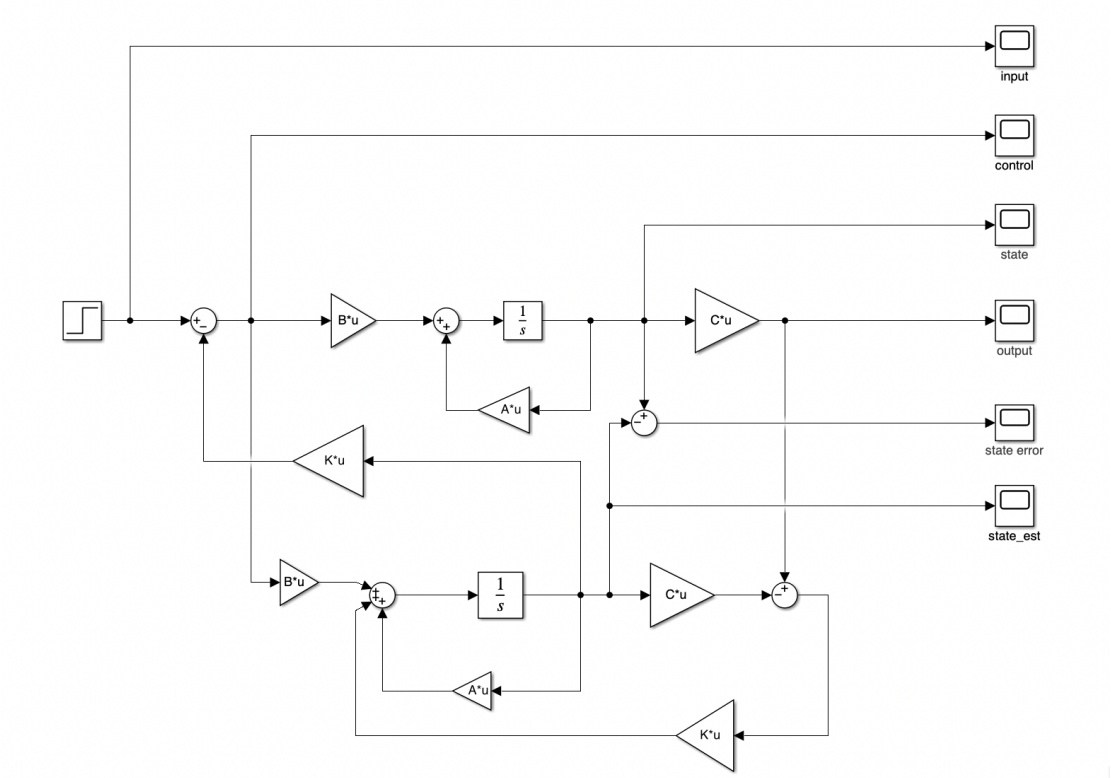

Figure 4-2 Observer control module

The observer is designed using pole placement method. The state estimate error and output which are non-zero initial state with zero external inputs are show as following.

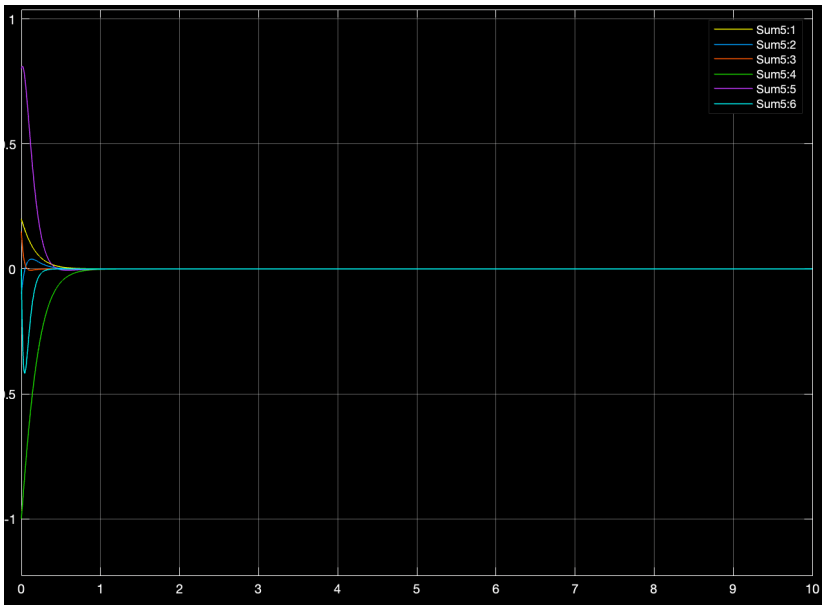

Figure 4-3 State estimate error

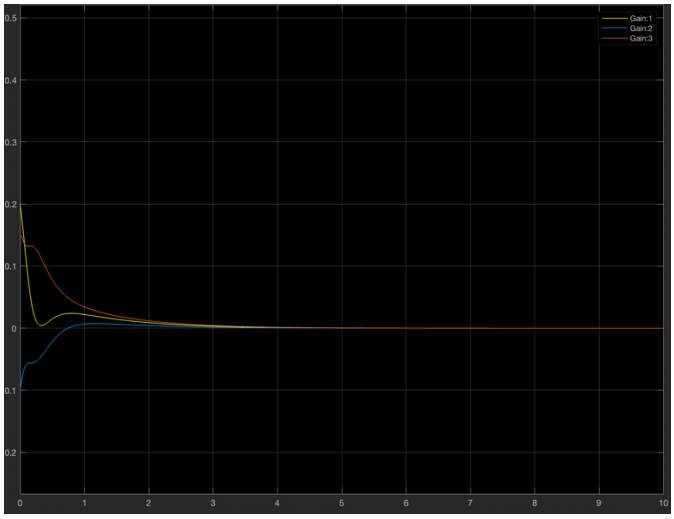

Figure 4-4 Observer outputs

The step responses of the three outputs of the pole placement observer and LQR method designed system meet the design specifications

4.3 Analysis

TABLE 4-2 Change the pole placement in observer

Pole	S1	S2	S3	S4	S5	S6
original	-8	-8	-12	-12	-24	-24
Change 1					-16	-16
Change 2					-32	-32

1) Decrease the absolute value of S5 and S6 to -16

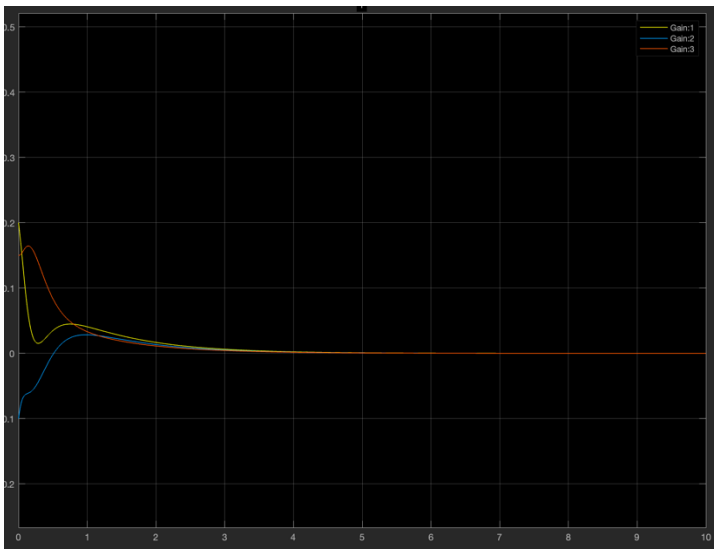

Figure 4-5 Observer outputs

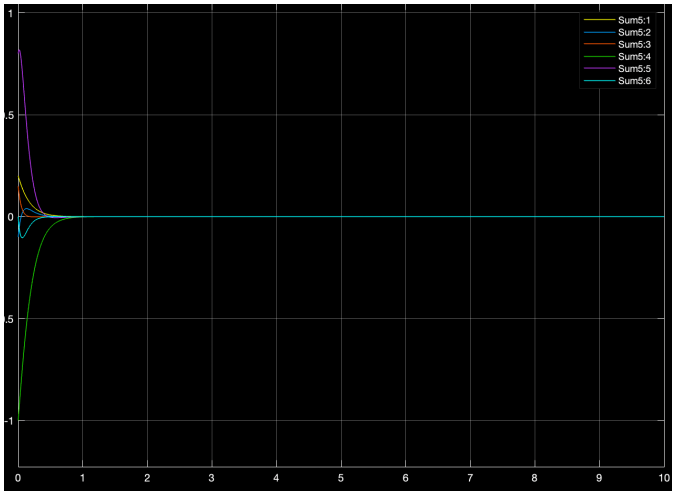

Figure 4-6 Observer state error

2) Increase the absolute value of S5 and S6 to -32

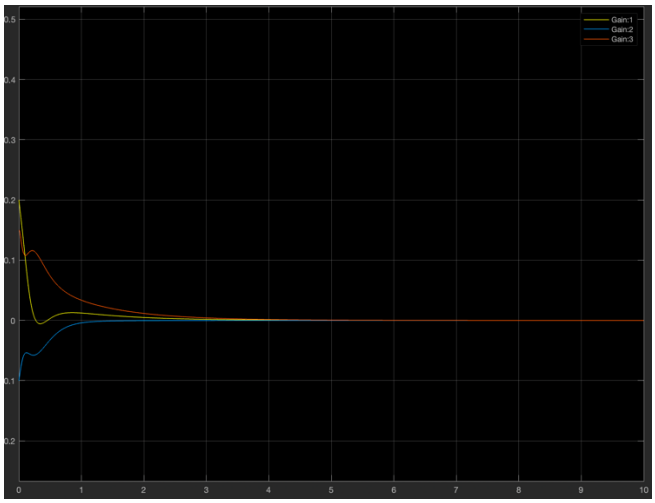

Figure 4-7 Observer output

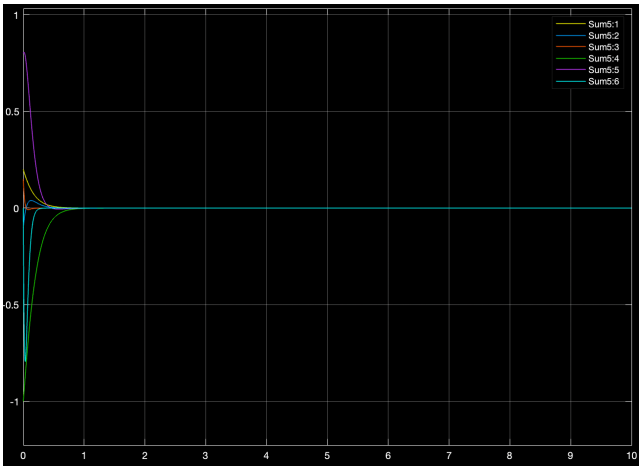

Figure 4-8 Observer state error

Discussion:

If the absolute value of the poles is larger, the system overshoot will be smaller, settling time will be shorter and state estimate error will be smaller. If the absolute value of the poles is smaller, the system overshoot will be larger, settling time will be longer and state estimate error will be larger.

5. Decoupling Controller

5.1 Controller Design

Since only two outputs $d(t)$ and $\varphi(t)$ are interested. C matrix needs to be designed as

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5-1}$$

The closed loop system is

$$\begin{aligned} \dot{x} &= (A - BK)x + BFr \\ y &= Cx \end{aligned} \tag{5-2}$$

The transform function can be written as

$$G(s) = C(Is - A)^{-1}B \tag{5-3}$$
$$H(s) = G(s)[I - K(Is - A)^{-1}B]F$$

On the other hand,

$$C_1^T AB = [27, 11.2], \quad C_2^T AB = [40, 60] \tag{5-4}$$

G(s) can be written as

$$G(s) = \begin{bmatrix} s^{-\sigma_1} & 0 \\ 0 & s^{-\sigma_2} \end{bmatrix} [B^* + C^*(sI - A)^{-1}B] \tag{5-5}$$

With $\sigma_1 = \sigma_2 = 2$. I set the placed poles:

$$\begin{cases} s^2 + 20s + 40 = 0 \\ s^2 + 50s + 49 = 0 \end{cases} \tag{5-6}$$

Then the matrix F and K can be calculated

$$F = (B^*)^{-1} = \begin{bmatrix} 0.055 & -0.104 \\ -0.037 & 0.024 \end{bmatrix} \tag{5-7}$$

$$K = (B^*)^{-1}C^* = \begin{bmatrix} 2.16 & 0.39 & -1.06 & 0.36 & 0 & -0.40 \\ -1.36 & -0.32 & 1.52 & -0.24 & 0 & 0.93 \end{bmatrix}$$

Since that the calculating the eigen value and find s3 = 2.5. The system is not stable.

5.2 Simulation

The initial state is x0, I use Simulink to simulate the system. My simulate module is showed in the following figure 5-1.

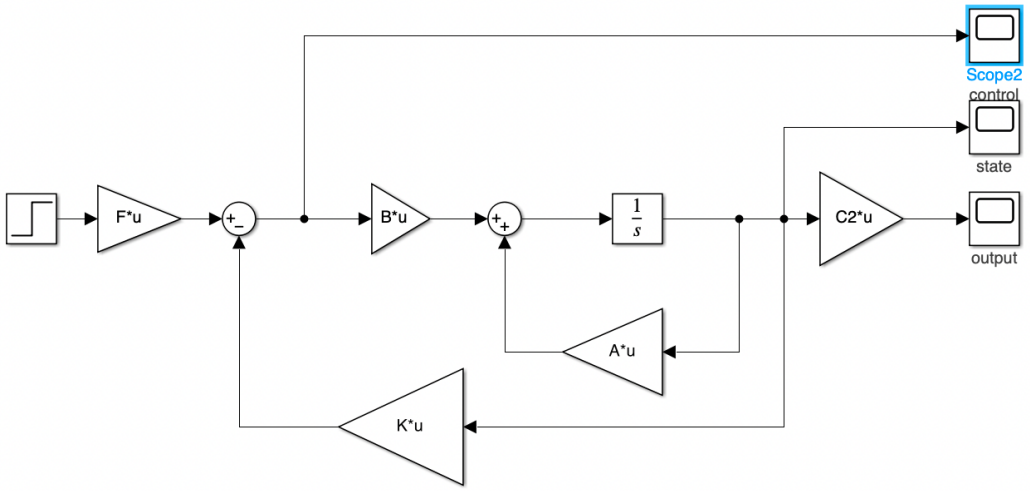

Figure 5-1 The design of the decoupling model

The poles are designed as formula 5-6 shows. The step response of the zero initial state:

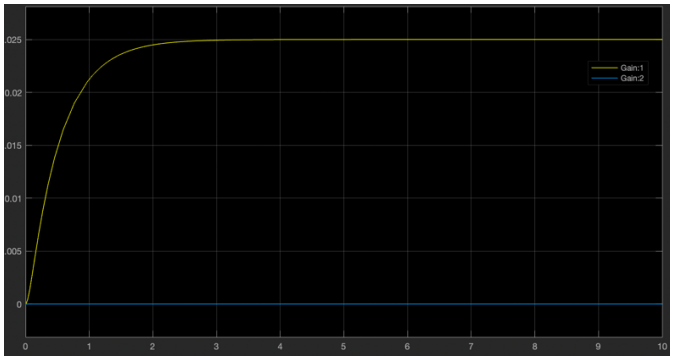

Figure 5-2 The step response of the decoupling model, r=[1,0]

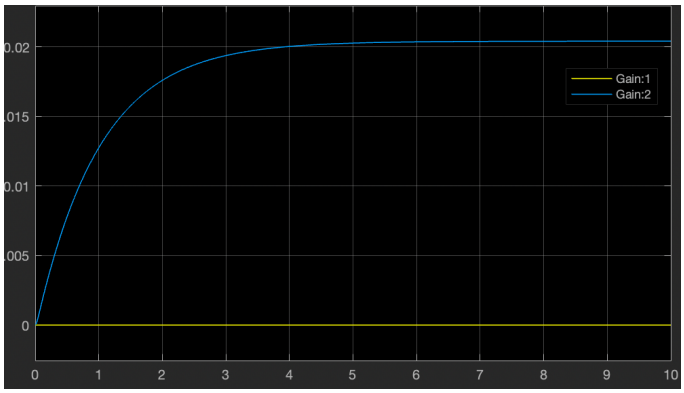

Figure 5-3 The step response of the decoupling model, r=[0,1]

The step response of the non-zero initial state(x0):

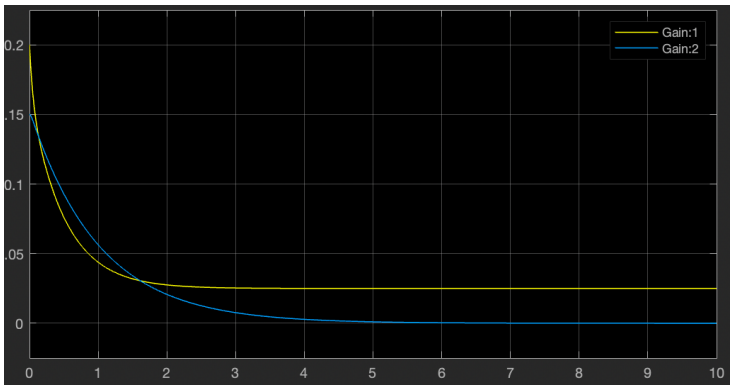

Figure 5-4 The step response of the decoupling model, r=[1,0]

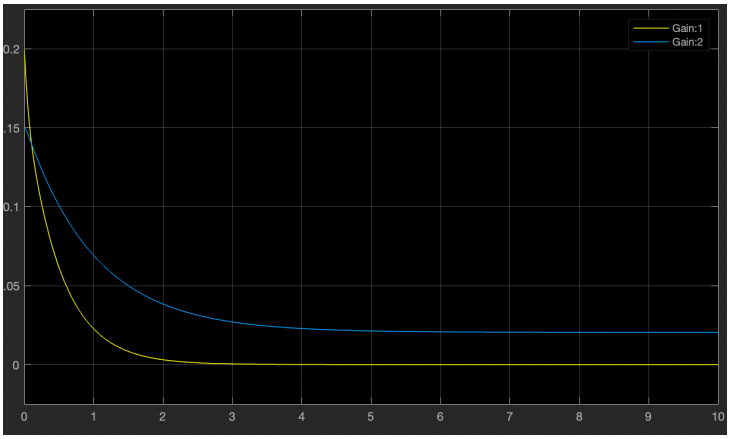

Figure 5-5 The step response of the decoupling model, r=[0,1]

The output, control signal and state response of all six which are non-zero initial state with zero external inputs are show as following.

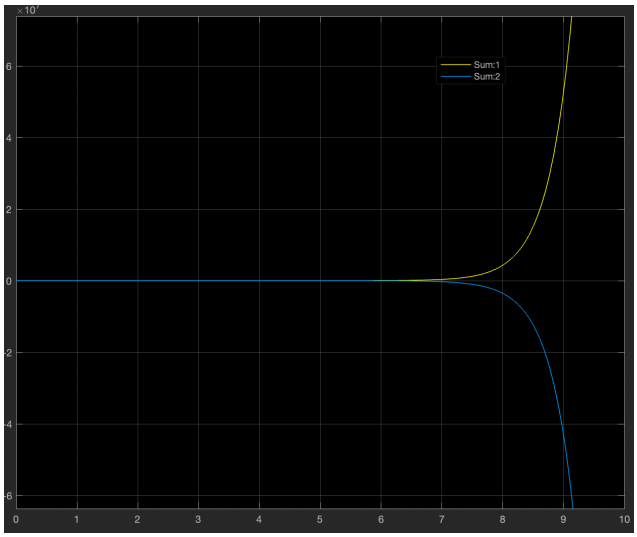

Figure 5-6 The control signal of the decoupling model

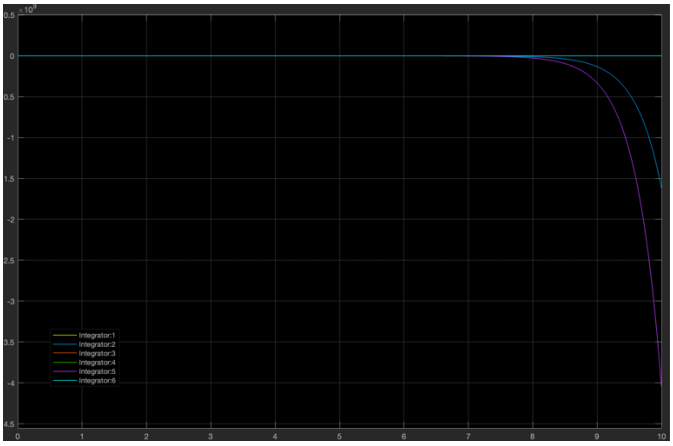

Figure 5-7 The all responses of the decoupling model

From the figure 5-6 and 5-7, the model is not stable.

6. Servo Controller

6.1 Controller Design

According to my student ID, 'A0260074M', the y_{sp} is calculated first,

$$y_{sp} = -\frac{1}{10} CA^{-1}B \begin{bmatrix} -0.5 + \frac{a-b}{20} \\ 0.1 + \frac{b-c}{a+d+10} \end{bmatrix} = \begin{bmatrix} 2.73 \\ 2.58 \\ 1.51 \end{bmatrix} \tag{6-1}$$

Since I only have three cheap sensors, there are 3 outputs can be measured. And the step disturbance for the two inputs, $w = [-1,1]^T$ which takes effect from time $t_d = 10s$ afterwards.

$$\begin{aligned} v(t) &= \int_0^t e(\tau) d\tau \\ \dot{v}(t) &= e(t) = r - y(t) = r - Cx(t) \end{aligned} \tag{6-2}$$

Thus I can form an augmented system

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} &= \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} B \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ I \end{bmatrix} r \\ \dot{\bar{x}} &= \bar{A}\bar{x} + \bar{B}u + \bar{B}w + \bar{B}_r r \\ y &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \end{aligned} \tag{6-3}$$

Then verify the rank of the controllability matrix

$$rank \begin{pmatrix} A & B \\ -C & 0 \end{pmatrix} = 8$$

```
>> rank(Qc)

ans =

     8
```

Figure 6-1 The result in MATLAB: rank of the controllability matrix

So, the augmented system is controllable. Then I use LQR to get the feedback gain K. The matrices Q and R are designed as $Q = diag(10,10,10,10,10,10,10,10,10), R = diag(1,1)$. Finally, K can be obtained.

33.4332	-29.3314	-10.4587	4.2975	-4.0811	0.5832	-0.3034	1.2806	-2.8754
-38.3698	38.7301	20.8314	-3.2482	5.7690	3.5212	-2.1148	-2.2226	-0.7667

Figure 6-2 The result in MATLAB: matrix K

$$u = -K\bar{x} = -\begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \tag{6-4}$$

Since the state of system cannot monitor directly, I design an observer based on LQR to estimate full-order observer. The observer can be expressed as:

$$\dot{\hat{x}} = A\hat{x} + Bu + Bw + L[y - \hat{y}]$$
$$\hat{y} = C\hat{x} \quad (6\text{-}5)$$

The poles of the controller are the eigenvalue of $A - BK_1$ is: s1 = -240, s2 = -50.5, s3 = -7.31, s4 = -2.62+1.18i, s5 = -2.62-1.18i, s6 = -1.74. Meanwhile, the Q and R in observer system is $\bar{Q} = diag(5,5,5,5,5,5), \bar{R} = diag(1,1,1)$. Finally, L can be calculated

2.1760	-0.5329	-0.1007
-0.5329	6.4912	-0.0199
-0.1007	-0.0199	2.2590
0.0144	2.1423	-1.3130
-6.7590	18.7097	1.8364
0.8772	-1.9573	0.0569

Figure 6-3 The result in MATLAB: L

6.2 Simulation

The initial state is x0, I use Simulink to simulate the system. My simulate module is showed in the following figure 6-4.

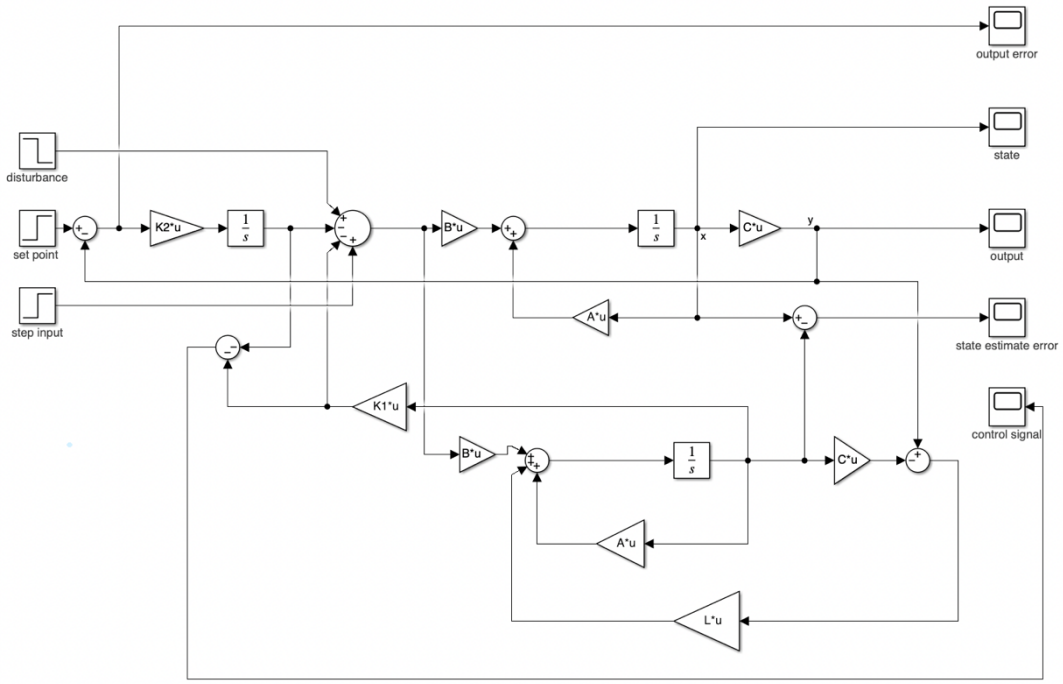

Figure 6-4 The design of servo system

The output, control signal, steady estimate error and state estimate error which are non-zero initial state with zero external inputs are show as following.

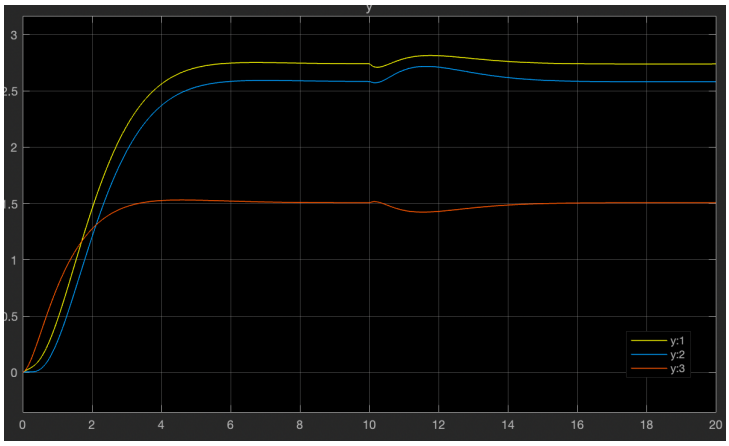

Figure 6-5 The output of servo system

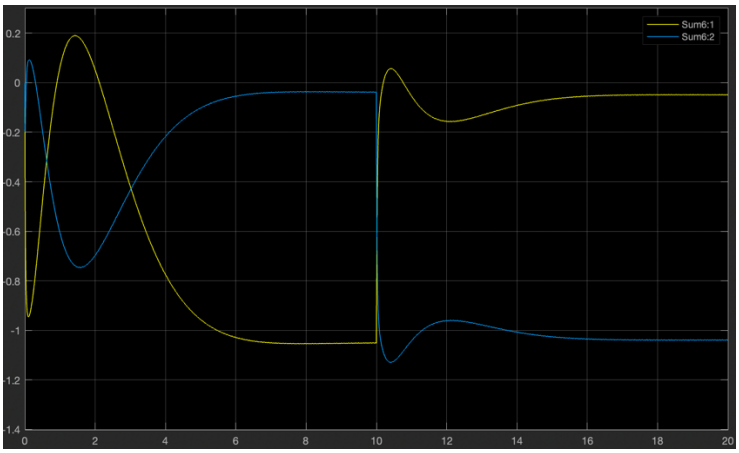

Figure 6-6 The control signal of servo system

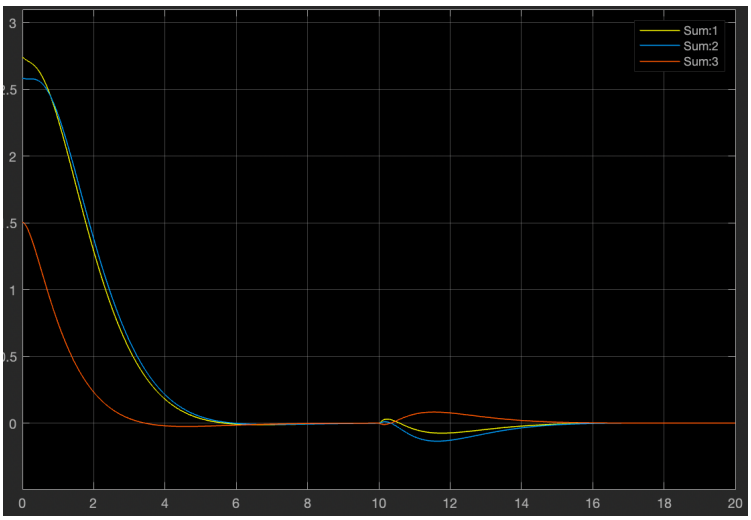

Figure 6-7 The steady error of servo system

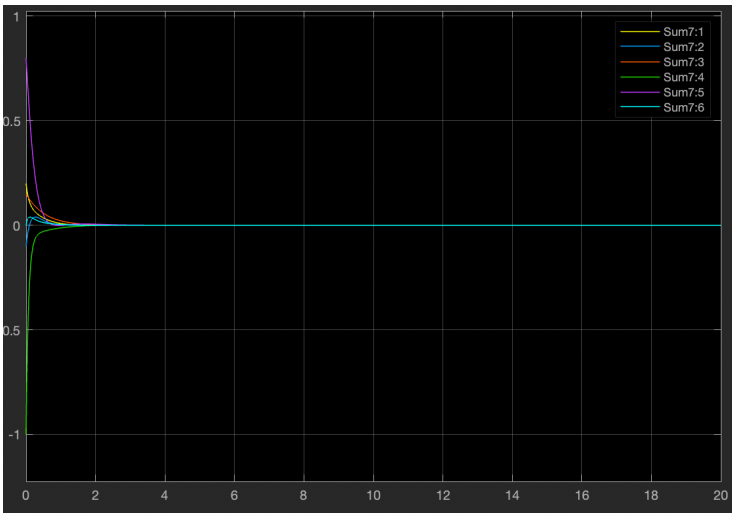

Figure 6-8 The state estimate error of servo system

It is obvious that the augmented system achieve zero state estimation error and steady-state error very quickly even disturbance appears. It can keep in the stable situation when disturbance appears.

7. Arbitrary set point

Conclusion: With zero steady-state error, three output cannot be maintained at an arbitrary constant set point.

7.1 Mathematical Analysis

I am going to explore and explain the problem of an arbitrary constant set point with zero steady-state error in two inputs and 3 outputs system. Since $\dim(u) = 2 < \dim(y) = 3$, it is impossible to maintain the three outputs at an arbitrary constant set point with zero steady-state error.
To achieve zero steady-state error, the system should follow the conditions:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A - BK_1 & -BK_2 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ I \end{bmatrix} r \tag{7-1}$$

The augment matrix is stable, and the augment system is controllable.
Let

$$Au = \begin{bmatrix} A - BK_1 & -BK_2 \\ -C & 0 \end{bmatrix} \tag{7-2}$$

And calculate the eigen value s of Au, s1 = -240, s2 = -50.48, s3 = -7.21, s4 = -3.63, s5 $\approx$ 0, s7 = -0.86 + 0.5i, s8 = -0.86 + 0.5i, s9 = -0.87 + 0.48i, s9 = -0.87 – 0.48i.

$$rank \begin{pmatrix} sI - A & B \\ -C & 0 \end{pmatrix} = n + p \tag{7-3}$$

The (7-3) exists if it is possible. Where n is the dimension of x and p is the dimension of y. However, formula (7-3) is equal to 8 not 9 after calculation in MATLAB.

Figure 7-1 The rank result in MATLAB

$$rank(C) = p \; and \; p > q \tag{7-4}$$

Where q is the dimension of u. Since n+p=9 and rank (7-3) is 8, it is impossible to maintain the three outputs at an arbitrary constant set point with zero steady-state error.

7.2 Simulation

Change the value of y_{sp}

TABLE 7-1 Arbitrary: changes of the set points

original	2.7391	2.5820	1.5060
Change 1	10	20	30
Change 2	0.03	0.02	0.01

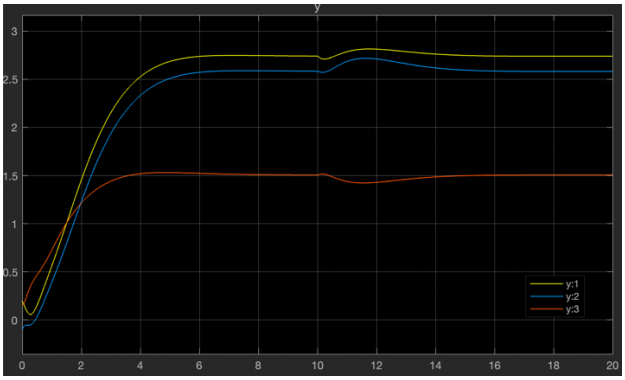

Figure 7-2 The output of original set points

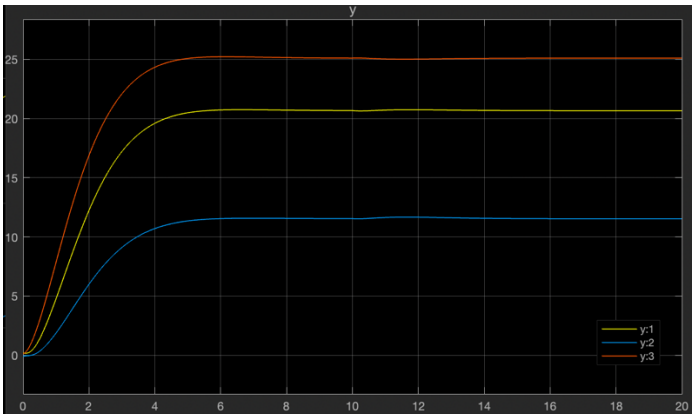

Figure 7-3 The output of change 1 set points

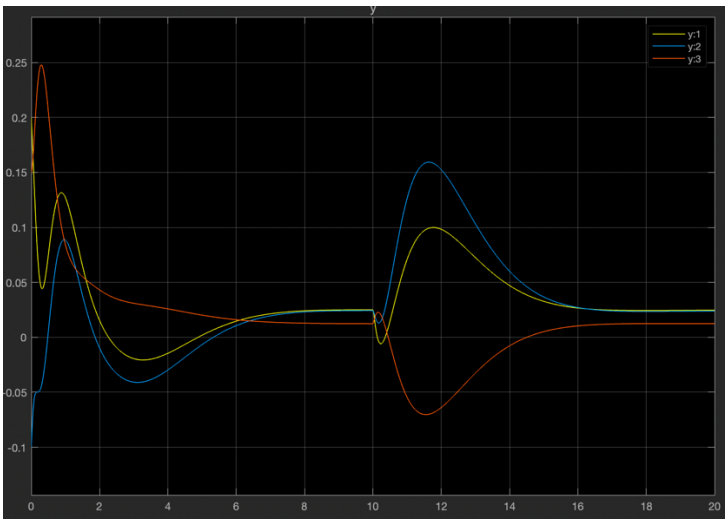

Figure 7-4 The output of change 2 set points

The result of Simulink prove the conclusion again. The system is not able to maintained at an arbitrary constant set point.

8. Conclusion

In this mini-project, I design the controller via different methods to satisfy the design specifications. First, I introduce the aim and requirement of the project.

In task 1, I use pole placement method to design the controller and discuss the effects of the position of the poles on system performance. If the absolute value of real part of the poles is larger than the point of original pole, the system settling time will be shorter and the input signal will be smaller. If the absolute value of imaginary part of the poles is larger than the point of original pole, the system overshoot will be larger and appear more oscillatory response.

In task 2, I use LQR method design the controller and discuss the effects of the weighting Q and R on system performance. If the value increases in Q, the settling time of the system will be shorter, and the input signal will be larger. If the value increases in R, the settling time of the system will be longer, and the input signal will be smaller.

In task 3, I design a state observer controller based on pole placement method and simulate the resultant LQR system. After investigating the effects of observer poles, I draw a conclusion. If the absolute value of the poles is larger, the system overshoot will be smaller, settling time will be shorter and state estimate error will be smaller.

In task 4, I design a decoupling controller with closed-loop stability. Since that there is one eigenvalue is positive, the system is not stable.

In task 5, I design a servo controller to achieve the set point. Because there are only 3 censors to measure output, I design a state observer using LQR method. The system shows strong ability to keep stable when disturbance appears.

In task 6, I try to explain the conclusion of mine about the arbitrary set point. The answer is no and I give a brief proof as well as an example.

From this project, I get a deeper understanding of the control system since I have not learned this field. From the practical design task for a control system, there are difficult still did not tackle in this mini-project. It is not a easy thing to satisfy the requirement, but there are many new modern control methods I do not know. In the future, based on the work I learned for this course, I would like to explore and focus on more powerful controller design methods.

Reference

[1] H. Satoh and T. Namerikawa, “Modeling and Robust Attitude Control of Stationary Self-sustaining Two-wheeled Vehicle,” p. 6.
[2] H. Satoh and T. Namerikawa, “Robust Stabilization of Running Self-Sustaining Two-wheeled Vehicle,” p. 6.

Appendix

Function: get the parameter of mine

```
function output = get_parameter(studentID)
% get studentID: string
% output parameter matrix

    % default studentID is mine
    if nargin < 1
        studentID = 'A0260074M';
    end

    % matriculation number
    a = str2num(studentID(5));
    b = str2num(studentID(6));
    c = str2num(studentID(7));
    d = str2num(studentID(8));

    % Physical parameters
    Mf = 2.14+c/20;
    Mr = 5.91-b/10;
    Mc = 1.74;
    LFf = 0.05;
    Lr = 0.128;
    Lc = 0.259;
    Jx = 0.5+(c-d)/100;
    alpha = 15.5-a/3+b/2;
    gamma = 11.5+(a-c)/(b+d+3);

    Hf = 0.18;
    Hr = 0.161;
    Hc = 0.098;
    LF = 0.133;
    LR = 0.308+(a-d)/100;
    mux = 3.33-b/20+a*c/60;
    beta = 27.5-d/2;
    delta = 60+(a-b)*c/10;

    den = Mf*Hf*Hf+Mr*Hr*Hr+Mc*Hc*Hc+Jx;
    g = 9.8;
    a51 = -1*Mc*g/den;
    a52 = (Mf*Hf+Mr*Hr+Mc*Hc)*g/den;
    a53 = (Mr*Lr*LF+Mc*Lc*LF+Mf*LFf*LR)*g/((LR+LF)*den);
    a54 = -1*Mc*Hc*alpha/den;
    a55 = -1*mux/den;
    a56 = Mf*Hf*LFf*gamma/den;
    b51 = Mc*Hc*beta/den;
    b52 = -1*Mf*Hf*LFf*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 -1*alpha 0 0;
        a51 a52 a53 a54 a55 a56;
        5 -3.6 0 0 0 -1*gamma];

    B=[0 0;
        0 0;
        0 0;
        beta 11.2;
        b51 b52;
        40 delta];

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

    C2 = [1 0 0 0 0 0;
        0 0 1 0 0 0];

    x0=[0.2;-0.1;0.15;-1;0.8;0];

    D=[0 0;0 0;0 0];

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

    output = {A,B,C,C2,x0,D,y_sp};

end
```

Task 1

```
clc
clear
studentID = 'A0260074M';
parameters = get_parameter();

syms s;
ts= 5 ;%settling time
mp=0.1;%overshoot

ep_min=abs(log(mp)/sqrt(pi^2+(log(mp))^2));
ep=0.8;
wn=4/(ep*ts);

lamda1=-ep*wn+wn*sqrt(1-ep^2)*1i*2;
lamda2=-ep*wn-wn*sqrt(1-ep^2)*1i*2;
lamda3=- 4;
lamda4=- 4;
lamda5=- 6;
lamda6=- 6;

polynomial1=1/(lamda1*lamda2)*(s-
lamda1)*(s-lamda2);
polynomial2=1/(lamda1*lamda2*lamda3*lamda4*
lamda5*lamda6)*(s-lamda1)*(s-lamda2)*(s-
lamda3)*(s-lamda4)*(s-lamda5)*(s-lamda6);
Ad_cof=double(coeffs(polynomial1));
Ad_cof2=double(coeffs(polynomial2));

figure();
step(tf(1,fliplr(Ad_cof2)))
hold on
step(tf(1,fliplr(Ad_cof)))
legend('add extra pole','original system')
xlim([0,10]);

clc
clear
studentID = 'A0260074M';
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
x0 = parameters{5};

%% reference model
syms s;
ts=5;%settling time
mp=0.1;%overshoot

ep_min=abs(log(mp)/sqrt(pi^2+(log(mp))^2));
ep=0.8;
wn=4/(ep*ts);

% lamda1=-ep*wn+wn*sqrt(1-ep^2)*1i;
% lamda2=-ep*wn-wn*sqrt(1-ep^2)*1i;
lamda1=-0.8+0.3i;
lamda2=-0.8-0.3i;
lamda3=- 3;
lamda4=- 3;
lamda5=- 2;
lamda6=- 2;

polynomial1=(s-lamda1)*(s-lamda2);
polynomial2=(s-lamda1)*(s-lamda2)*(s-
lamda3)*(s-lamda4)*(s-lamda5)*(s-lamda6);
polynomial3 = (s-lamda1)*(s-lamda2)*(s-
lamda3);
polynomial4 = (s-lamda4)*(s-lamda5)*(s-
lamda6);

Ad_cof1=double(coeffs(polynomial1));
Ad_cof2=double(coeffs(polynomial2));
Ad_cof3=double(coeffs(polynomial3));
Ad_cof4=double(coeffs(polynomial4));
K_verify = place(A,B,[lamda1 lamda2 lamda3
lamda4 lamda5 lamda6]);

%% Full Rank pole placement

syms k11 k12 k13 k14 k15 k16 k21 k22 k23
k24 k25 k26 ;
K=[k11 k12 k13 k14 k15 k16;
      k21 k22 k23 k24 k25 k26];

% controllability matrix
W=[B A*B A^2*B A^3*B];
% verify if W is controlible
```

```
assert(rank(W(:,1:6))==6);

Cmatirx =
[B(:,1),A*B(:,1),A^2*B(:,1),B(:,2),A*B(:,2)
,A^2*B(:,2)];
inv_Cmatrix=inv(Cmatirx);
bcol1=3;
bcol2=3;

T=[inv_Cmatrix(bcol1,:);
      inv_Cmatrix(bcol1,:)*A;
      inv_Cmatrix(bcol1,:)*A^2;
      inv_Cmatrix(bcol1+bcol2,:);
      inv_Cmatrix(bcol1+bcol2,:)*A;
      inv_Cmatrix(bcol1+bcol2,:)*A^2];

A_bar=T*A/(T);
B_bar=T*B;
A_bar(abs(A_bar)<10^(-10))=0;
B_bar(abs(B_bar)<10^(-10))=0;

Am=A_bar-B_bar*K;
Ad=[0 1 0 0 0 0 ;
        0 0 1 0 0 0;
        -Ad_cof3(1:3) 0 0 0;
        0 0 0 0 1 0;
        0 0 0 0 0 1;
        0 0 0 -Ad_cof4(1:3)];

 %% solve rotation
rotation=Am==Ad;
K_num=solve(rotation);
K_ans=struct2array(K_num);
K_ans=double(K_ans);
K_ba=[K_ans(1:6);
    K_ans(7:12)];
K_calculated=K_ba*T;

%% Plot the performence

t=0:0.02:10;
Af=A-B*K_calculated;
sys=ss(Af,B,C,0);
```

```
len = size(t,2);
u0=zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];

% zero inputs and x0 initial state
[y,tout,x]=lsim(sys,u0,t,x0);

figure()
plot(t,x)
grid on
legend('x1','x2','x3','x4','x5','x6')
xlabel('time')
ylabel('state')
title('zero inputs and x0 initial state')

figure()
plot(t,y)
grid on
legend('y1','y2','y3')
xlabel('time')
ylabel('output')
title('zero inputs and x0 initial state')

for i = 1:length(t)
% calculate the u
u_0(i,:) = -K_calculated*x(i,:)';
end
figure()
plot(t,u_0)
grid on
legend('uc','uh')
xlabel('time')
ylabel('control signal')
title('zero inputs and x0 initial state')

figure()
step(sys);
grid on
```

Task 2

```
clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};

%% system design

% verify controbility
W=[B A*B A^2*B A^3*B];
assert(rank(W(:,1:6))==6);

% LQR method
Q=[1 0 0 0 0 0
   0 10 0 0 0 0
   0 0 100 0 0 0
   0 0 0 100 0 0
   0 0 0 0 10 0
   0 0 0 0 0 1]*1;
R=[100 0
   0 1];

%[K1,~,P]=lqr(A,B,Q,R)

gamma=[A -B/R*B';-Q -A'];
[eig_vector,eig_value]=eig(gamma);
eig_value_sum=sum(eig_value);
vueigen=eig_vector(:,real(eig_value_sum)<0)
;
P=vueigen(7:12,:)/vueigen(1:6,:);
K_calculated=real(inv(R)*B'*P);

%% PLOT figure
t=0:0.01:10;
Af=A-B*K_calculated;
sys=ss(Af,B,C,0);
len = size(t,2);
u0=zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];

% zero inputs and x0 initial state
[y,tout,x]=lsim(sys,u0,t,x0);

figure()
plot(t,x)
legend('x1','x2','x3','x4','x5','x6')
xlabel('time')
ylabel('state')
title('zero inputs and x0 initial state')

figure()
plot(t,y)
legend('y1','y2','y3')
xlabel('time')
ylabel('output')
title('zero inputs and x0 initial state')

for i = 1:length(t)
    u_in(i,:) = -K_calculated*x(i,:)';
end
figure()
plot(t,u_in)
legend('uc','uh')
xlabel('time')
ylabel('control signal')
title('zero inputs and x0 initial state')

clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};
```

```
%% evaluate LQR performance when changing Q
and R
Q1=[1 0 0 0 0 0
   0 1 0 0 0 0
   0 0 1 0 0 0
   0 0 0 1 0 0
   0 0 0 0 1 0
   0 0 0 0 0 1].*50;
%R=[1 0
   %0 1].*0.5;

Q2=[1 0 0 0 0 0
   0 1 0 0 0 0
   0 0 1 0 0 0
   0 0 0 1 0 0
   0 0 0 0 1 0
   0 0 0 0 0 1].*0.5;
R=[1 0
   0 1];

% change Q
gamma1=[A -B/R*B';-Q1 -A'];
gamma2=[A -B/R*B';-Q2 -A'];

[eig_vector1,eig_value1]=eig(gamma1);
eig_value1_sum=sum(eig_value1);
vueigen1=eig_vector1(:,real(eig_value1_sum)
<0);
P1=vueigen1(7:12,:)/vueigen1(1:6,:);
K1=inv(R)*B'*P1;

[eig_vector2,eig_value2]=eig(gamma2);
eig_value2_sum=sum(eig_value2);
vueigen2=eig_vector2(:,real(eig_value2_sum)
<0);
P2=vueigen2(7:12,:)/vueigen2(1:6,:);
K2=inv(R)*B'*P2;

%% PLOT figure
t=0:0.01:10;
Af1=A-B*K1;
Af2=A-B*K2;
sys1=ss(Af1,B,C,D);
sys2=ss(Af2,B,C,D);
len = size(t,2);
u0=zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];

[y1,tout1,x1]=lsim(sys1,u0,t,x0);
figure()
plot(t,x1)
legend('x1 Q=50I','x2 Q=50I','x3 Q=50I','x4
Q=50I')
xlabel('time')
ylabel('state')
grid on
figure()
plot(t,y1)
legend('out1 Q=50I','out2')
grid on

%hold on
[y2,tout2,x2]=lsim(sys2,u0,t,x0);
figure()
plot(t,x2)
legend('x1 Q=0.5I','x2 Q=0.5I','x3
Q=0.5I','x4 Q=0.5I')
xlabel('time')
ylabel('state')
grid on
figure()
plot(t,y2)
legend('out1 Q=0.5I','out2')
grid on
```

Task 3

```
clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};

%% resultant observer-based LQR system

Q=[1 0 0 0 0 0
   0 1 0 0 0 0
   0 0 1 0 0 0
   0 0 0 1 0 0
   0 0 0 0 1 0
   0 0 0 0 0 1]*10;
R=[1 0
   0 1];

gamma=[A -B/R*B';-Q -A'];
[eig_vector,eig_value]=eig(gamma);
eig_value_sum=sum(eig_value);
vueigen=eig_vector(:,real(eig_value_sum)<0)
;
P=vueigen(7:12,:)/vueigen(1:6,:);
K_calculated=real(inv(R)*B'*P);
Af=A-B*K_calculated;
[eig_vector_Af,eig_value_Af]=eig(Af);

sys=ss(Af,B,C,D);
orig_pole=pole(sys);

%% full order pole placement,
% A_ba = A', B_ba = C', K_ba = L'
syms s
dir_pole = [-8,-8,-12,-12,-32,-32];
polynomial1 = (s-dir_pole(1))*(s-
dir_pole(2));
polynomial2 = (s-dir_pole(3))*(s-
dir_pole(4));
polynomial3 = (s-dir_pole(5))*(s-
dir_pole(6));
Ad_cof1=double(coeffs(polynomial1));
Ad_cof2=double(coeffs(polynomial2));
Ad_cof3=double(coeffs(polynomial3));

syms l11 l12 l13 l14 l15 l16 l21 l22 l23
l24 l25 l26 l31 l32 l33 l34 l35 l36 ;
L=[l11 l12 l13 l14 l15 l16;
    l21 l22 l23 l24 l25 l26;
    l31 l32 l33 l34 l35 l36];

W_bar=[C' A'*C' (A')^2*C' (A')^3*C'];
assert(rank(W_bar(:,1:6))==6);
Cmatrix=W_bar(:,1:6);
C_reconstructed=[Cmatrix(:,1),Cmatrix(:,4),
Cmatrix(:,2),Cmatrix(:,5),Cmatrix(:,3),Cmat
rix(:,6)];
inv_Cmatrix=inv(C_reconstructed);
bcol1=2;
bcol2=2;
bcol3=2;

T=[inv_Cmatrix(bcol1,:);
    inv_Cmatrix(bcol1,:)*A';
    inv_Cmatrix(bcol1+bcol2,:);
    inv_Cmatrix(bcol1+bcol2,:)*A';
    inv_Cmatrix(bcol1+bcol2+bcol3,:);
    inv_Cmatrix(bcol1+bcol2+bcol3,:)*A';];

Abar=T*(A')/(T);
Bbar=T*C';
Abar(abs(Abar)<10e-10)=0;
Bbar(abs(Bbar)<10e-10)=0;
Am=Abar-Bbar*L;
Ad=[0 1 0 0 0 0 ;
        -Ad_cof1(1:2) 0 0 0 0;
        0 0 0 1 0 0;
        0 0 -Ad_cof2(1:2) 0 0;
        0 0 0 0 0 1;
        0 0 0 0 -Ad_cof3(1:2)];
```

```
% solve rotation
rotation=Am==Ad;
L_number=solve(rotation);
L_answer=struct2array(L_number);
L_answer=double(L_answer);
Lbar=[L_answer(1:6);
    L_answer(7:12);
    L_answer(13:18)];
L_estimated=Lbar*T;
L_estimated=L_estimated';
L_estimated=real(L_estimated);
```

Task 4

```
clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};

%% decoupling performance

syms s

for i = 1:6
    if C2(1,:)*A^(i-1)*B ~= 0
        degree1 = i;
        break
    end
end
for i = 1:6
    if C2(2,:)*A^(i-1)*B ~= 0
        degree2 = i;
        break
    end
end

B_star = [C2(1,:)*A^(degree1-
1)*B;C2(2,:)*A^(degree2-1)*B];

Phi_A1=A^2+22*A+40*eye(6);

Phi_A2=A^2+50*A+49*eye(6);

C_star = [C2(1,:)*Phi_A1;C2(2,:)*Phi_A2];

F = inv(B_star);
K = F*C_star;
Bf=B*F;
Af = A-B*K;
decouple_model=ss(Af,Bf,C2,0);
W=[Bf Af*Bf Af^2*Bf Af^3*Bf];
assert(rank(W)==6);

p=pole(decouple_model);
H=C2*inv(s*eye(6)-Af)*Bf;

[~,eigenvalue] = eig(Af);

%% Plot
% non-zero inputs and zero initial state
figure()
step(decouple_model);
grid on

% zero inputs and x0 initial state
t=0:0.01:10;
len = size(t,2);
u0=zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];

[y,tout,x]=lsim(decouple_model,u0,t,x0);

figure()
plot(t,x)
grid on
legend('x1','x2','x3','x4','x5','x6')
xlabel('time')
ylabel('state')
title('zero inputs and x0 initial state')

figure()
plot(t,y)
grid on
legend('y1','y2')
xlabel('time')
ylabel('output')
title('zero inputs and x0 initial state')
```

Task 5

```
clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};

%% Servo control + LQR
w = [-1;1];

% verify controlbility
Qc = [A B;C zeros(3,2)];
assert(rank(Qc)==8);

A_bar=[A zeros(6,3);-C zeros(3,3)];
B_bar=[B;zeros(3,2)];
B_w_bar=[B;zeros(3,2)];
B_r_bar=[zeros(6,3);eye(3)];
C_bar=[C,zeros(3,3)];

Q=[1 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0;
   0 0 0 1 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 1 0 0 0;
   0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 1]*10;
R=[1 0
   0 1]*1;

gamma=[A_bar -B_bar/R*(B_bar');-Q -A_bar'];

[eig_vector,eig_value]=eig(gamma);
eig_value_sum=sum(eig_value);
vueogen=eig_vector(:,real(eig_value_sum)<0)
;
P=vueogen(10:18,:)/vueogen(1:9,:);
K_calculated=real(inv(R)*(B_bar')*P);

K1=K_calculated(:,1:6);
K2=K_calculated(:,7:9);

%% full order Observer LQR method
% A_ba = A', B_ba = C', K_ba = L'
Qbar=[1 0 0 0 0 0
      0 1 0 0 0 0
      0 0 1 0 0 0
      0 0 0 1 0 0
      0 0 0 0 1 0
      0 0 0 0 0 1]*5;
Rbar=[1 0 0
       0 1 0
       0 0 1]*1;

Phi1 = [A',-C'/Rbar*C;-Qbar,-A];
[eig_vector_observed,eig_value_observed]=ei
g(Phi1);
eig_value_observed_sum=sum(eig_value_observ
ed);
vueigen_observered=eig_vector_observed(:,re
al(eig_value_observed_sum)<0);
P_observed=vueigen_observered(7:12,:)/vueig
en_observered(1:6,:);
Kbar=real(inv(Rbar)*C*P_observed);
L=Kbar';
L=real(L);
```

Task 6

```
clc;
clear;
% get parameters
parameters = get_parameter();
A = parameters{1};
B = parameters{2};
C = parameters{3};
C2 = parameters{4};
x0 = parameters{5};
D = parameters{6};
ysp = parameters{7};

%% Servo control + LQR
w = [-1;1];

% verify controlbility
Qc = [A B;C zeros(3,2)];
assert(rank(Qc)==8);

A_bar=[A zeros(6,3);-C zeros(3,3)];
B_bar=[B;zeros(3,2)];
B_w_bar=[B;zeros(3,2)];
B_r_bar=[zeros(6,3);eye(3)];
C_bar=[C,zeros(3,3)];

Q=[1 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0;
   0 0 0 1 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 1 0 0 0;
   0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 1]*10;
R=[1 0
   0 1]*1;

gamma=[A_bar -B_bar/R*(B_bar');-Q -A_bar'];

[eig_vector,eig_value]=eig(gamma);
eig_value_sum=sum(eig_value);
vueogen=eig_vector(:,real(eig_value_sum)<0)
;
P=vueogen(10:18,:)/vueogen(1:9,:);
K_calculated=real(inv(R)*(B_bar')*P);

K1=K_calculated(:,1:6);
K2=K_calculated(:,7:9);

a11 = A-B*K1;
a12 = -B*K2;
Au = [a11,a12;-C,zeros(3,3)];
eigvalue = eig(Au);

for i=1:9
np11 = eigvalue(i).*eye(6,6)-A;
np = [np11,B;-C,zeros(3,2)];
a(i)=rank(np);
end

%% full order Observer LQR method
% A_ba = A', B_ba = C', K_ba = L'
Qbar=[1 0 0 0 0 0
      0 1 0 0 0 0
      0 0 1 0 0 0
      0 0 0 1 0 0
      0 0 0 0 1 0
      0 0 0 0 0 1]*5;
Rbar=[1 0 0
       0 1 0
       0 0 1]*1;

Phi1 = [A',-C'/Rbar*C;-Qbar,-A];
[eig_vector_observed,eig_value_observed]=ei
g(Phi1);
eig_value_observed_sum=sum(eig_value_observ
ed);
vueigen_observered=eig_vector_observed(:,re
al(eig_value_observed_sum)<0);
P_observed=vueigen_observered(7:12,:)/vueig
en_observered(1:6,:);
Kbar=real(inv(Rbar)*C*P_observed);
L=Kbar';
L=real(L);
```