Control of a Stationary Self-Balancing Two-wheeled Vehicle

Assignment For

EE5101/ME5401 Linear Systems

Submitted by

Jia Yansong A0263119H

Email: jiayansong@u.nus.edu

2022.11.13

Mechanical Engineering
Faculty of
National University of Singapore

Session 2022/2023

Abstract

Self-balancing two-wheeled vehicle is quite useful for people who have difficulty in riding a bicycle and it has been developing by applying modern control theory. In this paper, we construct a linear state-space system to simulate the true state of this vehicle in real world, and use pole-placement control, LQR control , controller with observer, decoupling control and integral control separately to control and simulate this system using MATLAB and Simulink. Controller and observer's performance on different inputs and disturbances are also included in this paper and final results prove that these control theories are suitable for this system.

Content

1. Introduction

It is quite difficult for a number of people to learn ride a bicycle since they can't balance themselves, so self-balancing two-wheeled vehicle is truly useful for them and it has a huge market potential.

A linear state-space model is construct to represent this system.

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

Because my matriculation number is A0263119H, so

$$a = 3, b = 1, c = 1, d = 9 \tag{2}$$

Put values to variables in $A, B\ and\ C$, get the system:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 6.5 & -10 & -15 & 0 & 0 \\ -25.904 & 22.333 & 7.2681 & -3.8857 & -5.0587 & 0.34894 \\ 5 & -3.6 & 0 & 0 & 0 & -11.654 \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 23 & 11.2 \\ 5.958 & -1.8025 \\ 40 & 60.2 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Control theories are proved useful to control the system to realize this vehicle self-balancing. Pole-placement control with full rank method and unity rank method, LQR control, LQR control with observer, decoupling control with state-feedback method, and integral control by set point tracking are proved stable for this system in following sections. In the final part, I prove that there are errors between actual outputs and set points if we set an arbitrary constant set point for this system. Models of different parts in Simulink and responses with different kinds of inputs are included respectively. Performance(overshot and settling time) of every controller and observer is tested satisfy with the design specifications.

The overshoot and 2% settling time of the design specifications are less than 10% and 5 seconds respectively.

$$\begin{cases} M_p = e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}} < 10\% \\ e^{-\xi w_n t_s} \approx 0.02 \end{cases} \tag{5}$$

$$\begin{cases} \xi = 0.6 \\ w_n \approx 1.32 \end{cases} \tag{6}$$

So the reference second order model's transfer function is

$$\frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} = \frac{1.74}{s^2 + 1.58s + 1.74} \tag{7}$$

The desired poles are

$$-0.79 + 1.056i\ and\ -0.79 - 1.056i. \tag{8}$$

MATLAB code *ME5401.m* file is the backbone of this paper which computes the key variables of controllers and observers, *place_pole.m*, *unity_place.m* and *my_lqr.m* files contain three main functions *place_pole, unity_pole,* and *my_lqr* used in *ME5401.m* file to compute state-feedback gain for designing controller and observers by full rank pole-placement method, unity rank pole-placement method and LQR method respectively. These MATLAB code files are all in appendix part.

2. Pole-placement Control

2.1 Full Rank Method

First compute the controllability matrix:

$$W_c = [B \quad AB \quad A^2B \quad A^3B \quad A^4B \quad A^5B] \tag{9}$$

Compute the rank of this matrix to check whether this system is controllable:

$$rank(W_c) = 6 \tag{10}$$

So this matrix is full rank and this system is controllable.

For MIMO system, next step is to select n independent vectors out of nm vectors form the controllability matrix in the strict order from left to right, and group them with the same input together in a square matrix C:

$$C = [b_1 \quad Ab_1 \quad A^2b_1 \quad b_2 \quad Ab_2 \quad A^2b_2] \tag{11}$$

Meanwhile, the index of input is:

$$d_1 = d_2 = 3 \tag{12}$$

Then compute T using $d_1, d_2 \; and \; C^{-1}$:

$$T = \begin{bmatrix} q_3^T \\ q_3^T A \\ q_3^T A^2 \\ q_6^T \\ q_6^T A \\ q_6^T A^2 \end{bmatrix} \tag{13}$$

Use T compute $\bar{A} \; and \; \bar{B}$:

$$\bar{A} = TAT^{-1} \qquad \bar{B} = TB \tag{14}$$

Construct $\overline{K}(2 \times 6)$ matrix:

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

Compare the non-trivial rows in $\bar{A} - \bar{B}\overline{K}$ matrix and A_d matrix. In this plant, the non-trivial rows are 3th and 6th row because $d_1 = 3, d_2 = 3, d_1 + d_2 = 6$. Then there are 12 equations and 12 variables, so there is only one solution of $\overline{K}$.

Finally, compute feedback gain K.

$$K = \overline{K}T \tag{16}$$

2.2 Unity Rank Method

First initialize q, which is a 2×1 vector in this plant. q must make the pair $\{A, Bq\}$ is controllable:

$$q = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix} \tag{17}$$

Second, use Ackermann formula to compute k^T:

$$k^T = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1] C^{-1} \phi_d(A) \tag{18}$$

Where

$$C^{-1} = [Bq \quad ABq \quad A^2Bq \quad A^3Bq \quad A^4Bq \quad A^5Bq] \tag{19}$$

Then according to the designed poles, μ_i, $i = 1,2,3,4,5,6$, (will be given value in experiment part), compute characteristic polynomial:

$$\phi_d(s) = \prod_{i=1}^{6} (s - \mu_i) \tag{20}$$

Finally, compute state feedback gain K.

$$K = qk^T \tag{21}$$

This method has infinity solutions for K, because the number of q is infinity.

2.3 Experiments and Results

After compute state-feedback gain by pole-placement methods, the state feedback law is $u = r - Kx$, so the closed-loop system is:

$$\dot{x} = (A - BK)x + Br \tag{22}$$

Construct system model in Simulink:

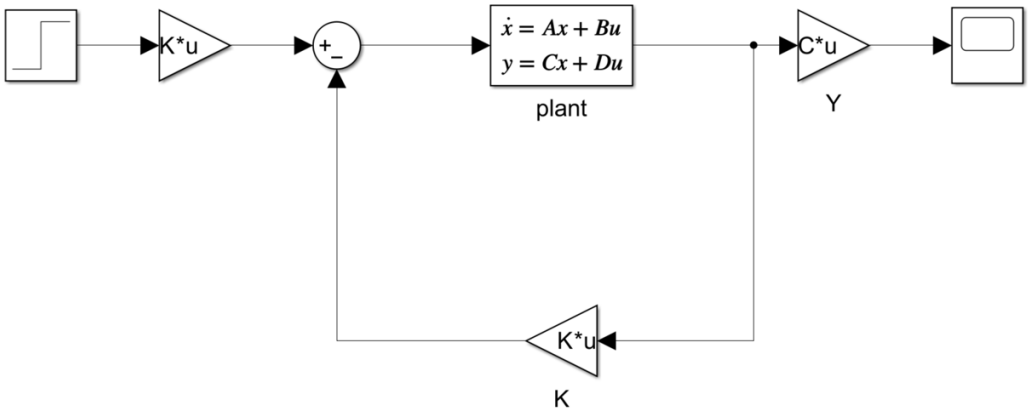

Figure 1 Model of Pole-Placement Method

Choose six poles:

(1) $poles = -1, -2, -3, -4, -5, -6$:

The r step response with zero initialization with Input(1) and Input(2) are showed in Figure 2 and Figure 3, control signal Kx is showed in Figure 4:

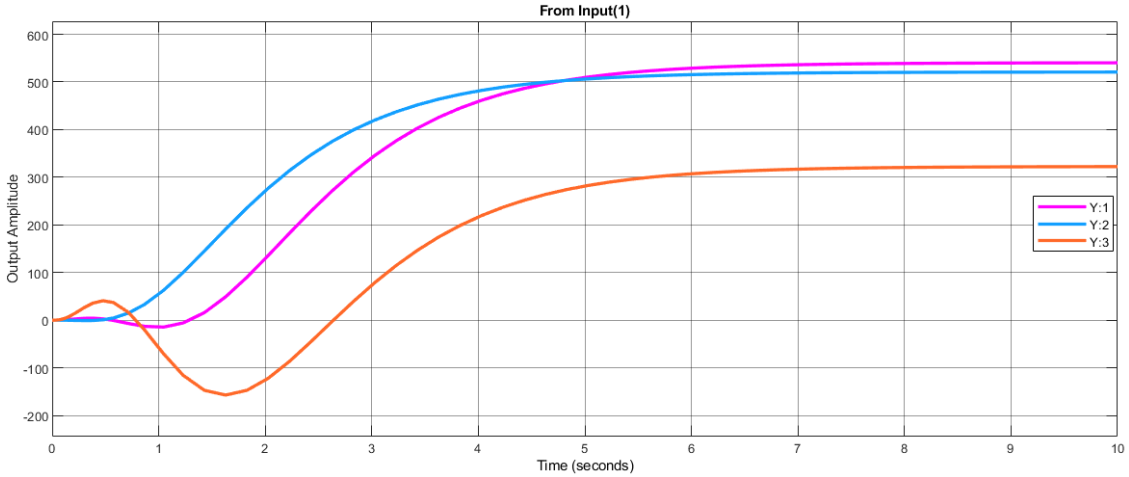

Figure 2 Input(1) Step Response

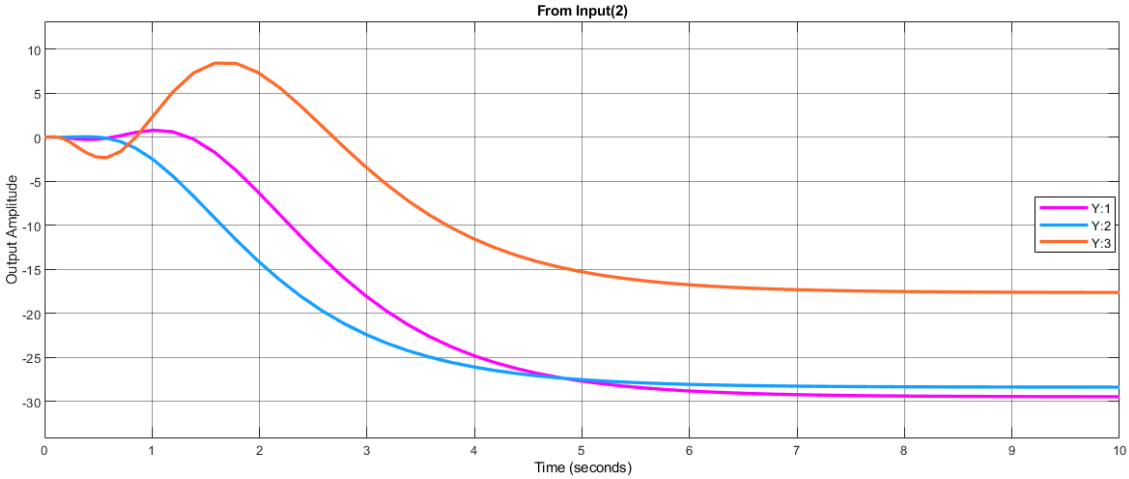

Figure 3 Input(2) Step Response

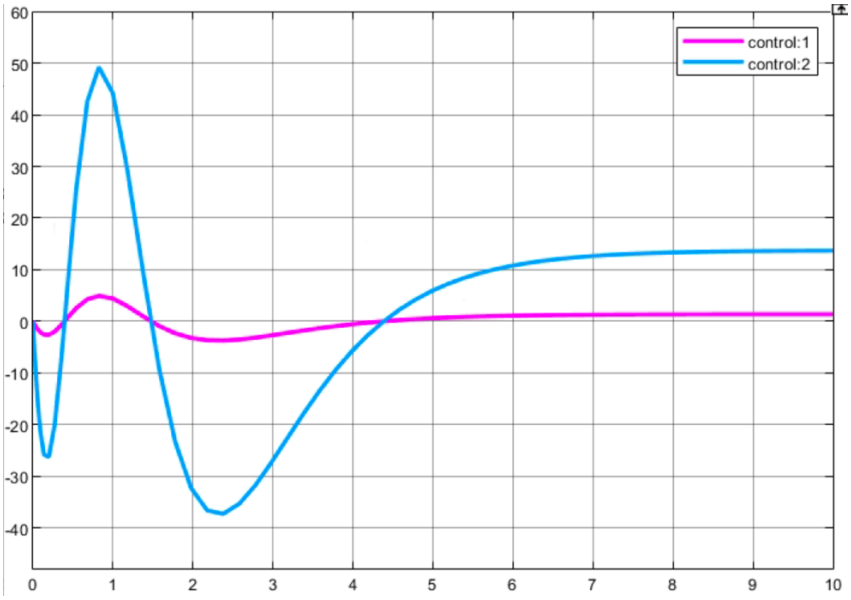

Figure 4 Control Signal of First Poles

(2) $poles = -2, -3, -4, -5, -6, -7$:

The r step response with zero initialization with Input(1) and Input(2) are showed in Figure 4 and Figure 5 and control signal Kx is showed in Figure 7:

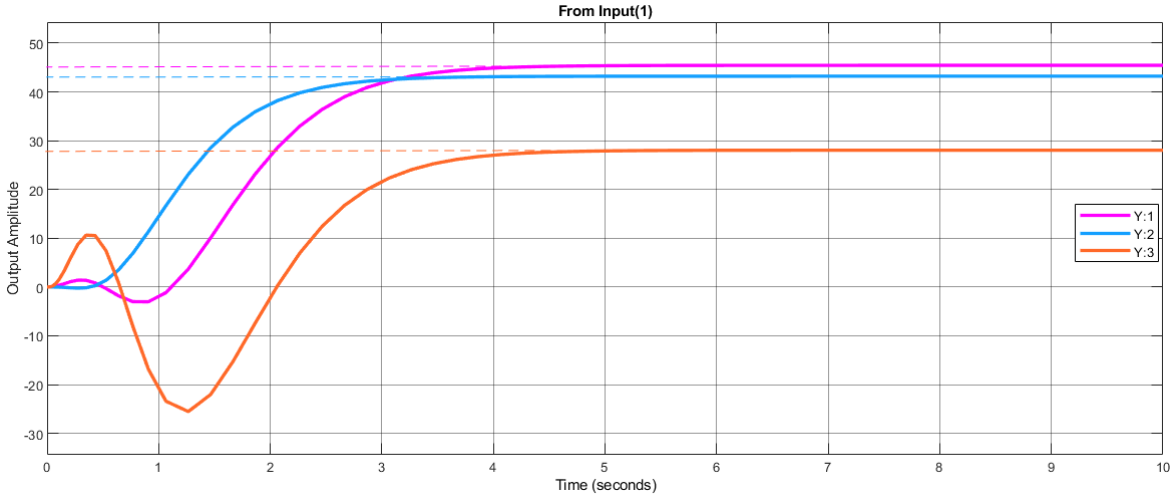

Figure 5 Input(1) Step Response

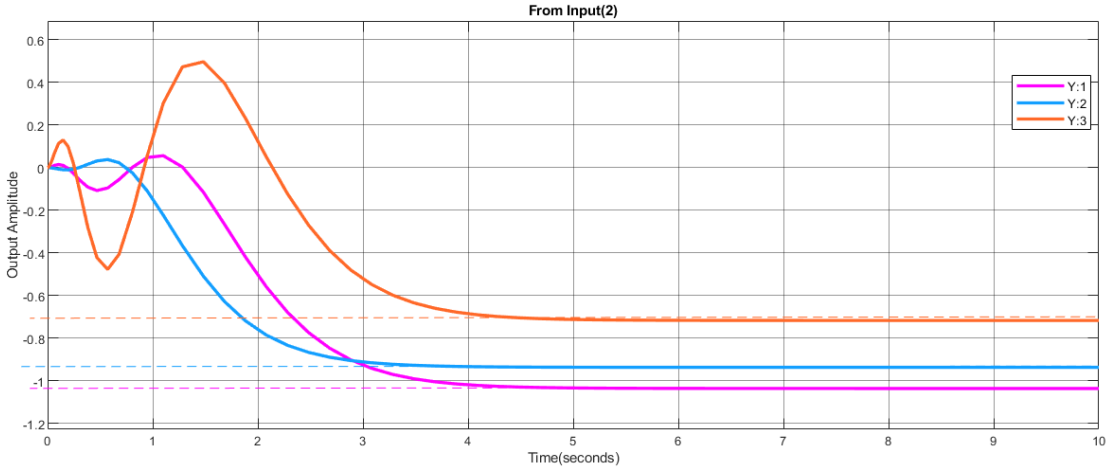

Figure 6 Input(2) Step Response

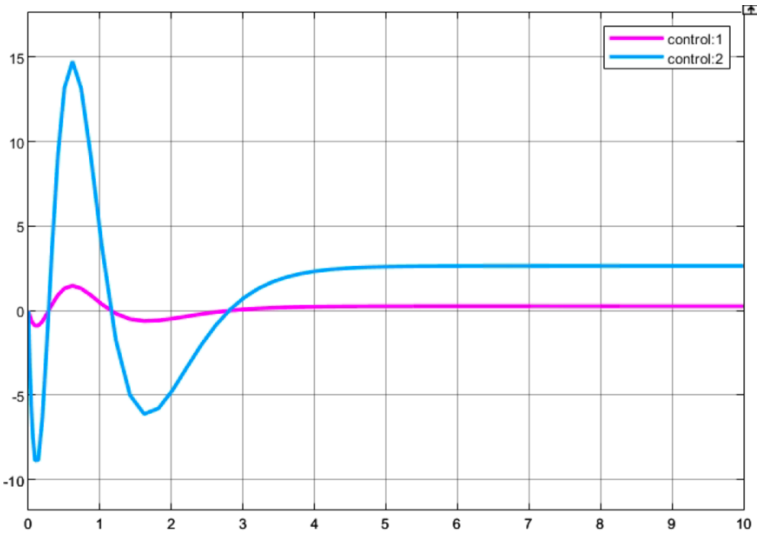

Figure 7 Control Signal of Second Poles

2.4 Analysis

First put the poles at $-1,-2,-3,-4,-5,-6$, but the three outputs' setting times of the closed-loop system are all bigger than 5s, but the overshoots of them are less than 10%, so put the poles further from the origin, when the poles are at$-2,-3,-4,-5,-6,-7$, the closed-loop system converges to steady state faster, so that the setting time is less than 5s and satisfies the design specifications. The control signal size of first poles are higher than second poles, so that the output amplitudes of first poles are also higher than those of second poles, because first poles need more time to converge to steady state but second poles can more quickly converge to be stable.

Finally, Poles: $-2,-3,-4,-5,-6,-7$ can satisfy all the design specifications, and don't cost much energy.

3. LQR Control

3.1 Method

The LQR optimal control is to find the control law that minimizes $J=\frac{1}{2}\int_0^\infty (x^TQx+u^TRu)dt$, and it turns out to be in the form of linear state feedback: $u=r-Kx$.

First is to find the positive definite solution P of the Riccati equation:

$$PA+A^TP-PBR^{-1}B^TP+Q=0 \tag{23}$$

Where A and B are plant parameters, Q and R are parameters given in LQR method at begin.

Step 1: Form the 12×12 matrix, then find its 6 stable eigenvalues.

$$\Gamma=\begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{24}$$

Step 2: Separate the selected stable eigenvectors (corresponding to stable eigenvalues) matrix to two parts:

$$\begin{bmatrix} v_i \\ \mu_i \end{bmatrix}, i=1,2,3,4,5,6 \tag{25}$$

Step 3: P is given by:

$$P=[\mu_1,\mu_2,\mu_3,\mu_4,\mu_5,\mu_6][v_1,v_2,v_3,v_4,v_5,v_6]^{-1} \tag{26}$$

After computing P from Riccati equation, K is given by:

$$K=R^{-1}B^TP \tag{27}$$

3.2 Experiments and Results

The model of this whole system in Simulink is showed in Figure 8:

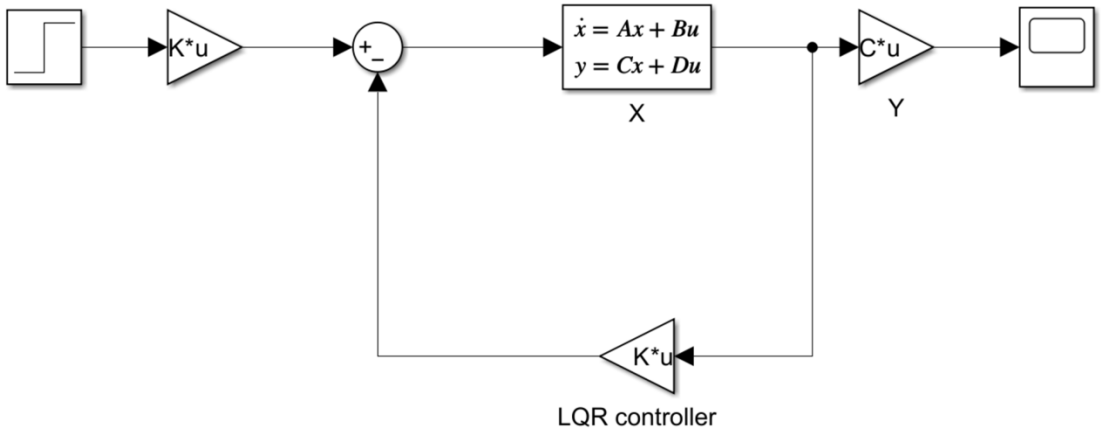

Figure 8 LQR Control model

Choose Q and R for LQR controller:

(1) $Q = diag(1,1,1,1,1,1)$ and $R = diag(1,1)$. The r step response with first input and second input are shown in Figure 9 and Figure 10. It is obviously that they don't satisfy the design specifications, because their outputs' overshoot and setting time are over 10% and 5 seconds.

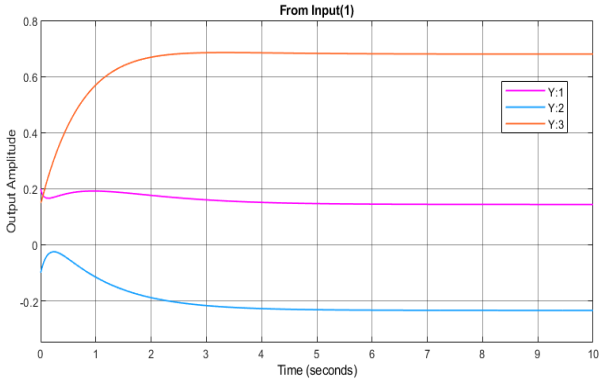

Figure 9 Input(1) Step Response of first QR pair

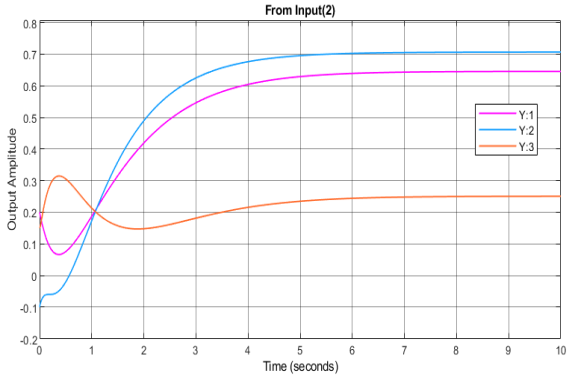

Figure 10 Input(2) Step Response of second QR pair

Control signal is shown in Figure 11:

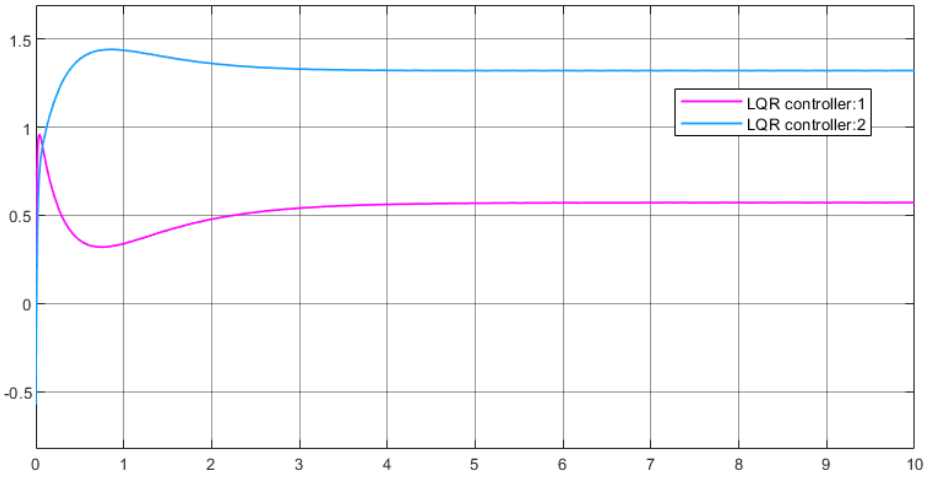

Figure 11 Control Signal of first QR pair

(2) $Q = diag(1, 10, 20, 1, 2, 3)$ and $R = diag(0.1, 0.1)$. The r step response with first input and second input are shown in Figure 12 and Figure 13. When I choose this pair of QR, it is obviously that the outputs are all satisfy the design specifications.

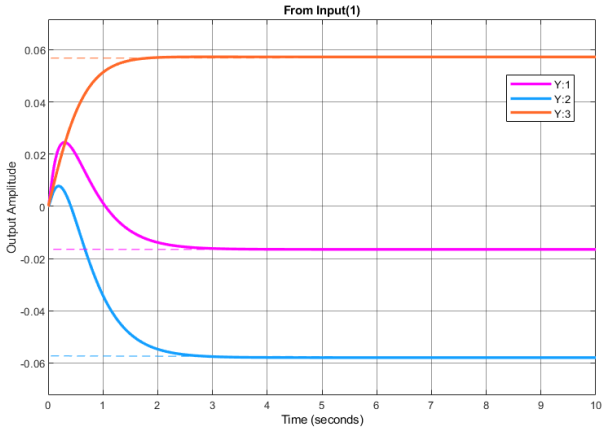

Figure 12 Input(1) Step Response of second QR pair

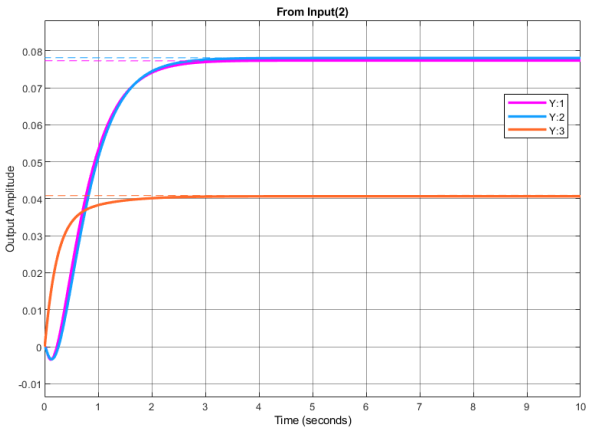

Figure 13 Input(2) Step Response of first QR pair

Control signal is shown in Figure 11:

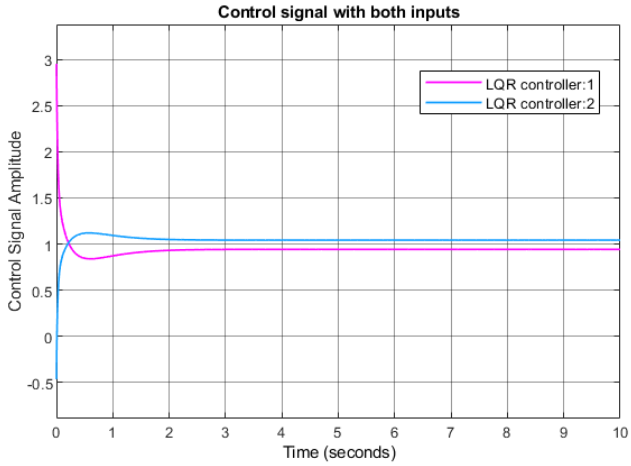

Figure 14 Control signal size of second QR pair

The state response with $x_0 = [0.2 \quad -0.1 \quad 0.15 \quad -1 \quad 0.8 \quad 0]^T$ and zero external input:

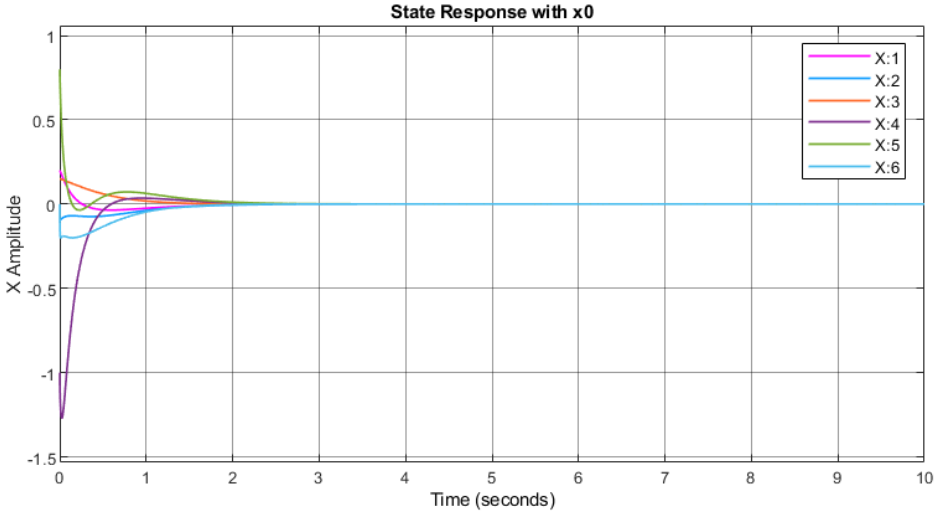

Figure 15 State Response with x0 and zero input

3.3 Analysis

When I set $Q = diag(1,1,1,1,1,1)\ and\ R = diag(1,1)$ first, LQR controller doesn't satisfy all the design specifications, when I set $Q = diag(1,10,20,1,2,3)\ and\ R = diag(0.1,0.1)$, the closed-loop system's performance satisfy all the requirements. And compare their control signal, the second pair's is larger than first pair's at begin, but smaller when the system converging to steady state, that's because the larger the elements of Q are, the faster the state variables converge to zero with larger control signal and larger energy cost, and on the other hand, the larger the elements of R, the slower response but smaller control signal size needed with smaller energy cost.

4. LQR Controller with Observer

4.1 Method

First check this plant whether is observable: construct observability matrix:

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix}, \qquad rank(O) = 6 \tag{28}$$

So it is observable.

Second, make $A - LC$ stable because $\dot{\tilde{x}} = (A - LC)\tilde{x}$, if $A - LC$ is stable, then the estimate error will converge to zero in steady state. By duality, because the pair (A, C) is observable, so $(\tilde{A}, \tilde{B}) = (A^T, C^T)$ is controllable.

Because $\det[sI - (A - LC)] = \det[sI - (\tilde{A} - \tilde{B}\tilde{K})]$, $where\ \tilde{A} = A^T, \tilde{B} = C^T, \tilde{K} = L^T$, it is the same as using pole-placement method to place desired stable poles for $A - LC$. Use the same function *place_pole* which is written previously in *place_pole.m* MATLAB code file to calculate the proper observer gain : $L = place\_pole(A', C', [desired\ poles])'$ by duality.

4.2 Experiments and Results

Model in Simulink:

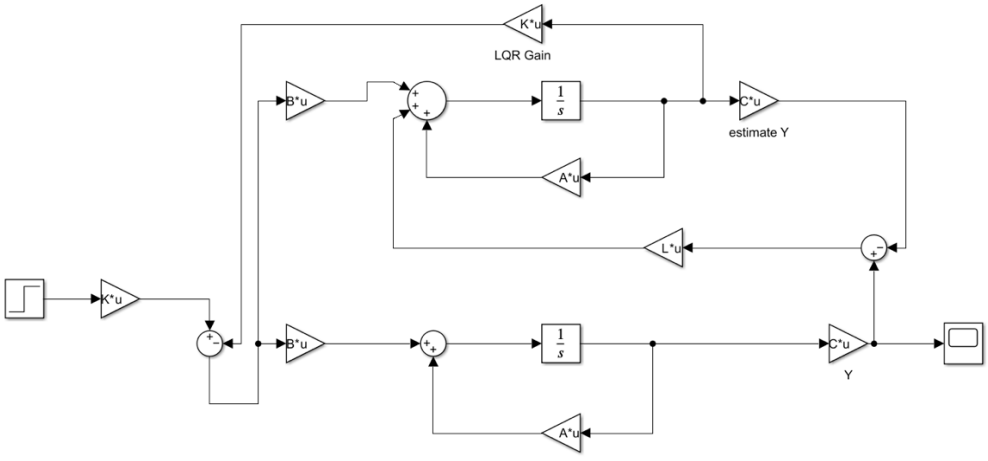

Figure 16 LQR Controller with Observer Model

It is suggested that place observer poles 3-5 times faster than the closed loop dominant poles designed by the controller. The initial states of observer are zeros, and that of plant is $[0.2 \quad -0.1 \quad 0.15 \quad -1 \quad 0.8 \quad 0]^T$, the LQR controller $Q = diag(1, 10, 20, 1, 2, 3)$ and $R = diag(0.1, 0.1)$.

Choose observer poles $[-5, -10, -15, -20, -25, -30]$:

$$L = place\_pole(A', C', [-5 \quad -10 \quad -15 \quad -20 \quad -25 \quad -30]) \tag{29}$$

$$L = \begin{bmatrix} 0 & 0 & 0 \\ -3.8857 & 29.9413 & 0.3489 \\ 0 & 0 & 43.3462 \\ 50 & 6.5 & -10 \\ -6.2479 & 170.8685 & 20.6282 \\ 5 & -3.6 & 244.8506 \end{bmatrix} \tag{30}$$

State estimate error:

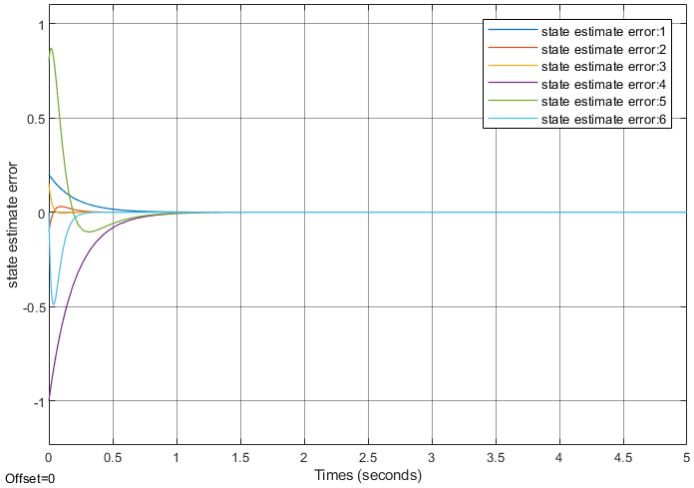

Figure 17 state estimate error

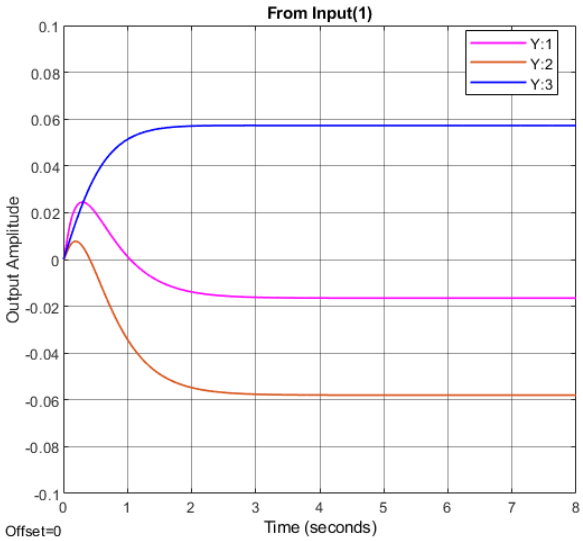

Figure 18 First Step Input Response

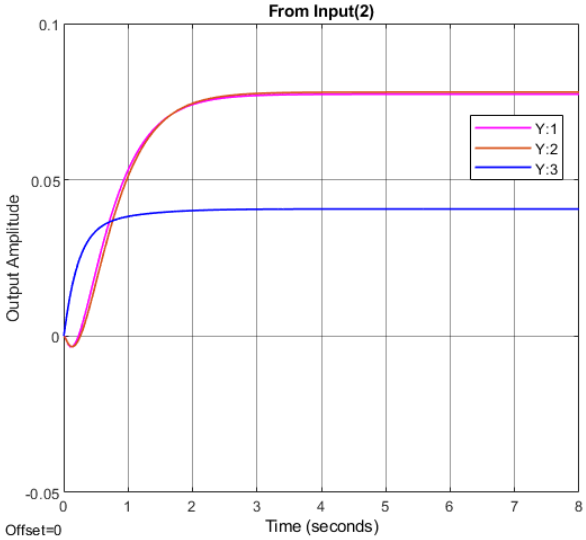

Figure 19 Second Step Input Response

4.3 Analysis

Comparing the controller performance with observer in this section with controller performance without observer, it is clear that at the begin, there is error between true states and estimated states, but with the closed control of observer, the error converges to zero fast and there in no difference on state and output between with and without observer in steady state.

5. Decoupling Control

5.1 Method

Decoupling control is to decouple a coupled plant, and it is usually required in practice for easy operations. There are two schemes to decouple a plant: state feedback with the state-space model and output feedback with the transfer function matrix.

This is a linear state-space model, so it is convenient to use first scheme, state feedback to decouple this plant. C is changed to $\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}$. The transfer function matrix:

$$G(s) = C(sI - A)^{-1}B \tag{31}$$

$$G(s) = \begin{bmatrix} g_1^T(s) \\ g_2^T(s) \\ \vdots \\ g_m^T(s) \end{bmatrix} \tag{32}$$

$$\sigma_i = \begin{cases} \min\left(j \middle| c_i^T A^{j-1}B \neq 0^T, j = 1,2,\dots,n\right); \\ n, \ \ if \ \ c_i^T A^{j-1}B = 0^T, j = 1,2,\dots,n \end{cases} \tag{33}$$

In this plant, $c_1^T B = [0 \quad 0], c_1^T AB = [23\ 11.2]\ and\ c_2^T B = [0 \quad 0], c_2^T AB = [40 \quad 60.2]$. So,

$$\begin{cases} \sigma_1 = 2, \ \ i = 1 \\ \sigma_2 = 2, \ \ i = 2 \end{cases} \tag{34}$$

$$B^* = \begin{bmatrix} c_1^T A^{\sigma_1 - 1}B \\ c_2^T A^{\sigma_2 - 1}B \end{bmatrix} = \begin{bmatrix} 23 & 11.2 \\ 40 & 60.2 \end{bmatrix} \tag{35}$$

In order to make the decoupled plant stable, design a stable $\phi_{f_i}(A) = (A + 2I)(A + 4I)$ for this plant. Then use Theorem 2, which shows

$$C^{**} = \begin{bmatrix} c_1^T \phi_{f_i}(A) \\ c_2^T \phi_{f_2}(A) \end{bmatrix} = \begin{bmatrix} c_1^T (A + 2I) \\ c_2^T (A + 4I) \end{bmatrix} = \begin{bmatrix} 4 & 6.5 & -10 & -11 & 0 & 0 \\ 5 & -3.6 & 16 & 0 & 0 & -3.6538 \end{bmatrix} \tag{36}$$

Then,

$$K = B^* C^{**} = \begin{bmatrix} 0.19731 & 0.46084 & -0.83408 & -0.70703 & 0 & 0.043693 \\ -0.048046 & -0.366 & 0.81999 & 0.46978 & 0 & -0.089727 \end{bmatrix} \tag{37}$$

$$F = B^{*-1} = \begin{bmatrix} 0.064275 & -0.011958 \\ -0.042708 & 0.024557 \end{bmatrix} \tag{38}$$

Finally, the plant is decoupled to this new system:

$$H = \begin{bmatrix} \frac{1}{(s+2)^2} & 0 \\ 0 & \frac{1}{(s+4)^2} \end{bmatrix} \quad (39)$$

5.2 Experiments and Results

Model of decoupling system in Simulink:

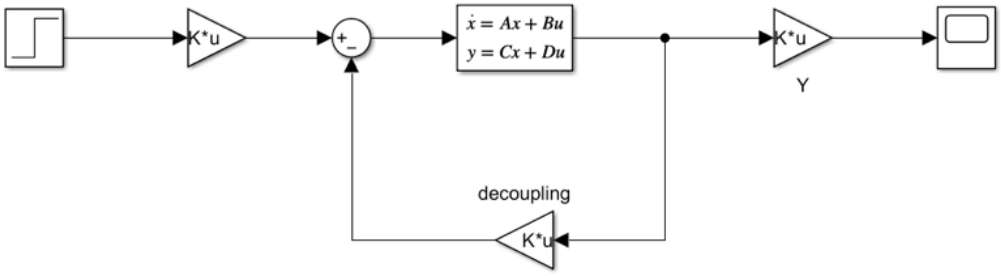

Figure 20 Decoupling system Model

Step input response of the decoupled system with zero initial states:

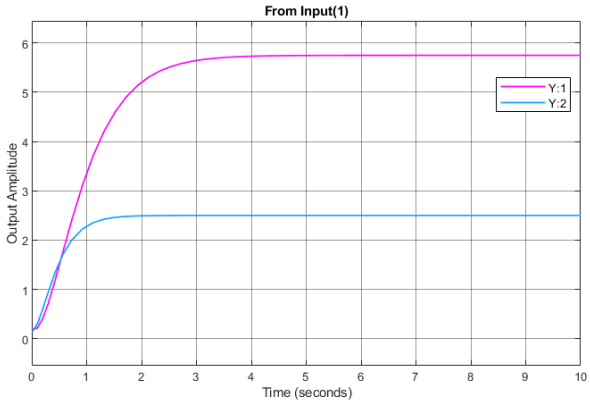

Figure 21Step Response of first Input

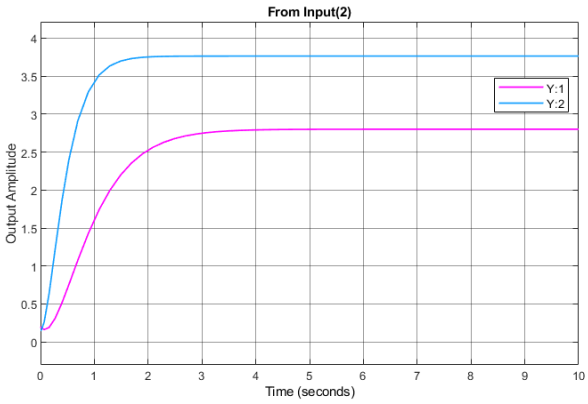

Figure 22 Step Response of second Input

Initial response with respect to x_0 and states with step input :

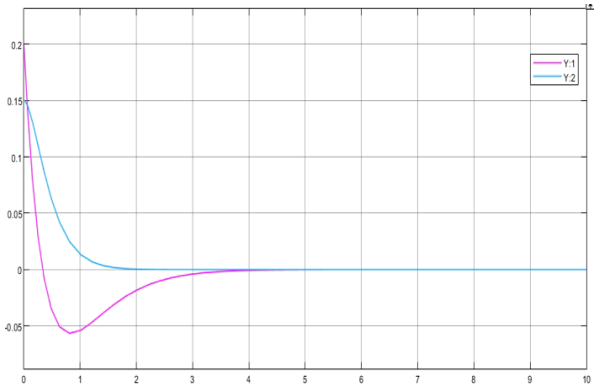

Figure 23 Initial response with x0

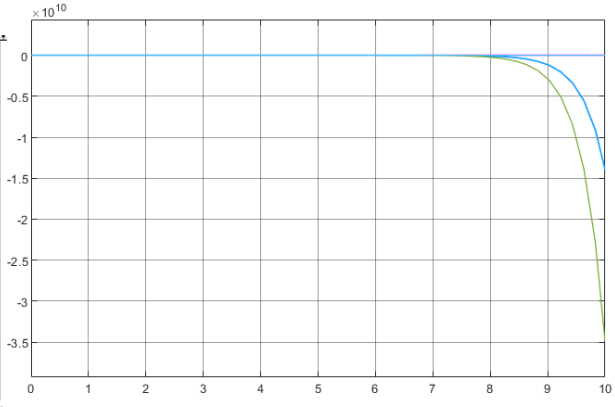

Figure 24 States with step input

5.3 Analysis

Compare with the Figure 21, 22 and 24, it is clear that the decoupled system is BIBO stable, but not internally stable, because the two outputs are all stable and satisfy the design specifications, but from Figure 24 we can see that states in this system are not stable, and go to infinity.

6. Integral Control

6.1 Method

It is given that $y_{sp} = -\frac{1}{10} C A^{-1} B \begin{bmatrix} -0.5 + \frac{a-b}{20} \\ 0.1 + \frac{b-c}{a+d+10} \end{bmatrix}$, and $a = 3, b = 1, c = 1, d = 9$, so

$$y_{sp} = \begin{bmatrix} 0.7147 \\ 0.7154 \\ 0.3842 \end{bmatrix} \tag{40}$$

Use servo mechanism:

$$\dot{v} = e(t) = r - y(t) = r - Cx(t) \tag{41}$$

Form augmented system:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A & O \\ -C & O \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} B \\ O \end{bmatrix} u + \begin{bmatrix} B_w \\ O \end{bmatrix} w + \begin{bmatrix} O \\ I \end{bmatrix} r \tag{42}$$

Check whether the augmented system is controllable or not:

$$rank \begin{pmatrix} A & B \\ -C & 0 \end{pmatrix} = 8 = n + m \tag{43}$$

Because the original plant is controllable, so the augmented system is controllable.
Use LQR control to control this augmented system, use compute LQR controller gain function *my_lqr*:

$$K = my\_lqr(\bar{A}, \bar{B}, Q, R) \tag{44}$$

Where

$$\bar{A} = \begin{bmatrix} A & O \\ -C & O \end{bmatrix}, \bar{B} = \begin{bmatrix} B \\ O \end{bmatrix}, Q = 3 \times eye(9), R = 0.1 \times eye(2) \tag{45}$$

There are only three cheap sensors to measure the output, so we need design an observer to estimate the states in this system to construct the closed-loop control system.

I just use the observer the same as the observer of LQR Controller with Observer part. Use function written previously:

$$L = place\_pole(A', C', [-5 \ -10 \ -15 \ -20 \ -25 \ -30])' \tag{46}$$

Finally, use this method, the output follows the reference input and can eliminate the disturbance.

6.2 Experiments and Results

Figure 25 shows the model of this control method in Simulink, and Figure 26, 27 show the output amplitude without disturbance and with disturbance taking effect from time $t_d = 10s$. Figure 28, 29 show the control signal with disturbance and estimate state error respectively.

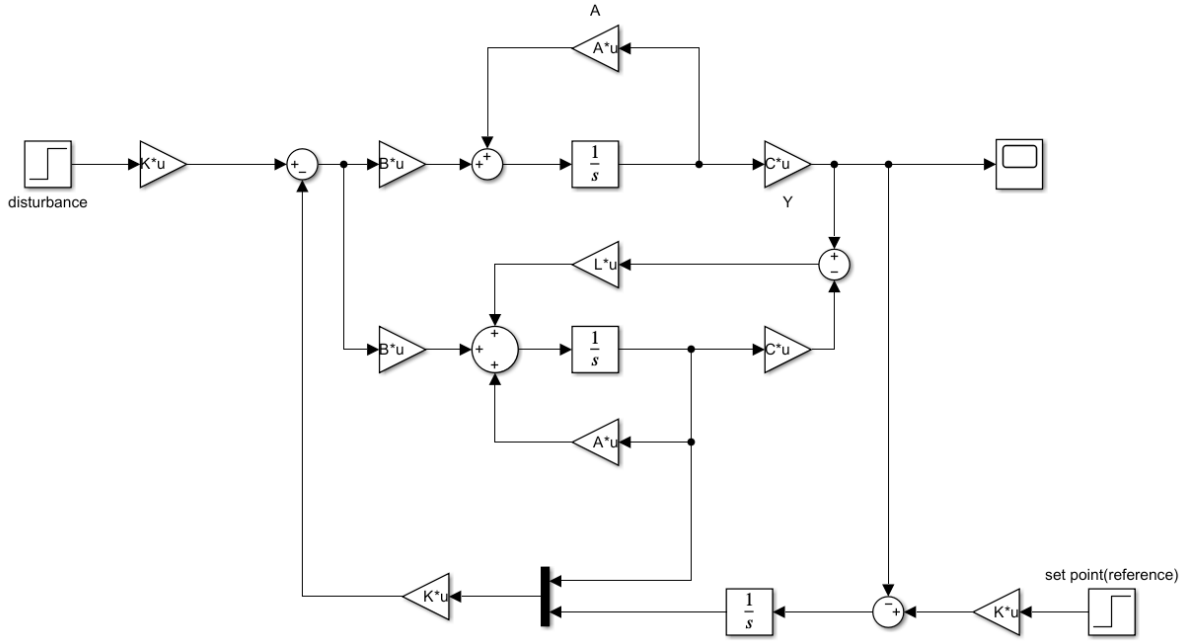

Figure 25 Model of Integral Control

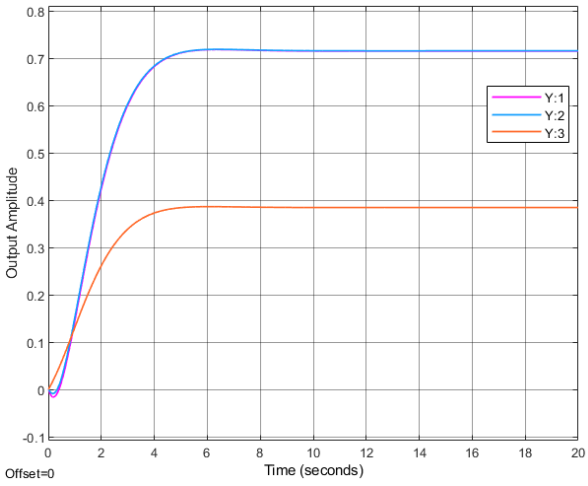

Figure 26 reference step response without disturbance

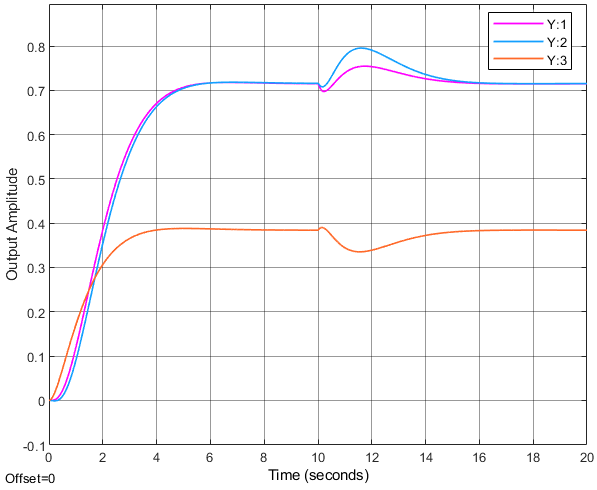

Figure 27 reference step response with disturbance

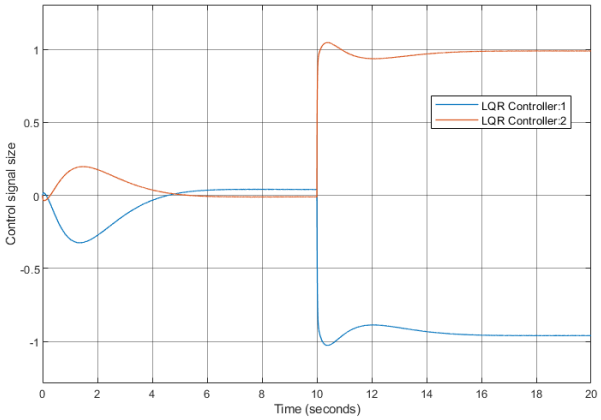

Figure 28 Control signal size with disturbance

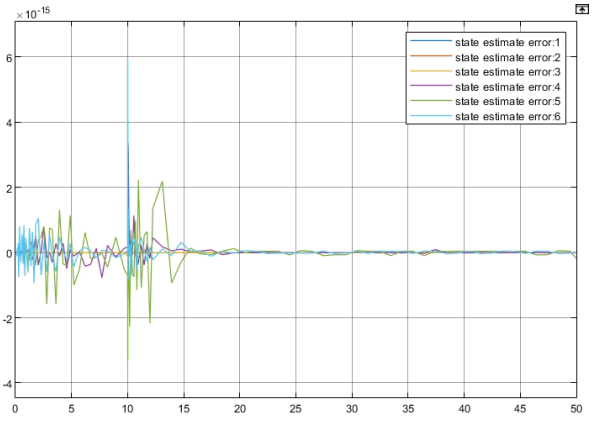

Figure 29 observer estimate state error

6.3 Analysis

Using servo control, we can control the output for reference tracking and eliminate disturbance at steady state.

7. Can we maintain the three outputs at an arbitrary constant set point with zero steady-state error?

For this system, there are two inputs and three outputs. When the system's states converge to stable by closed-loop LQR Controller, which is indicated in LQR Control part,

$$\dot{x} = 0 \tag{47}$$

So, with the control law: $u = -Kx + Fr$, without disturbance. And for computing and proving conveniently, because F and r are all the values designed ourselves, I use $u_d \in \mathbb{R}^{2\times1}$ to replace Fr, then the original linear system can be written as:

$$\begin{gathered} 0 = (A - BK)x + BFr = (A - BK)x + Bu_d \\ y = Cx \end{gathered} \tag{48}$$

So the output is:

$$y = -C(A - BK)^{-1}Bu_d \tag{49}$$

If we want to maintain the three outputs at an arbitrary constant set point, which is $y_{sp} \in \mathbb{R}^{3\times1}$, then there should be a corresponding input $u_d \in \mathbb{R}^{2\times1}$. From equation 47, then we can derive an equation set with two variables (two inputs) but three solutions (three outputs):

$$\begin{aligned} y_1 &= \gamma_1 u_{d1} + \gamma_2 u_{d2} \\ y_2 &= \gamma_3 u_{d1} + \gamma_4 u_{d2} \\ y_3 &= \gamma_5 u_{d1} + \gamma_6 u_{d2} \end{aligned} \tag{50}$$

So from equation 48, it is clear that we can't derive a exact solution $u_{d1}\ and\ u_{d2}$ because the number of equations is larger than that of variables. But we can still use Least-Squares Estimation method [1] to get the best solutions for u_{di}:

$$\phi = \begin{bmatrix} \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_4 \\ \gamma_5 & \gamma_6 \end{bmatrix}, u_d = \begin{bmatrix} u_{d1} \\ u_{d2} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{51}$$

$$\widehat{u_d} = (\phi^T\phi)^{-1}\phi^T Y \tag{52}$$

Where $\widehat{u_d}$ is the best solution for this equation set, but it can't be precise because it is estimated.

Conclusion:

So finally, there are errors between the ideal y_{sp} and the actual y which is the output of when we input this $\widehat{u_d}$ to the system. And we can't maintain the three outputs at an arbitrary constant set point with zero steady-state error.

Here is an numerical example:

I arbitrarily set $y_{sp} = \begin{bmatrix}1\\2\\3\end{bmatrix}$, use equation (52), we can derive $\widehat{u_d}$:

$$\widehat{u_d} = (\phi^T\phi)^{-1}\phi^T Y = \begin{bmatrix}0.5122 & -6.3433 & 11.1982\\ 5.7600 & 4.0760 & 5.7946\end{bmatrix}\begin{bmatrix}1\\2\\3\end{bmatrix} = \begin{bmatrix}21.4203\\31.2958\end{bmatrix} \quad (53)$$

Construct closed-loop control in Simulink as the same as LQR Control part:

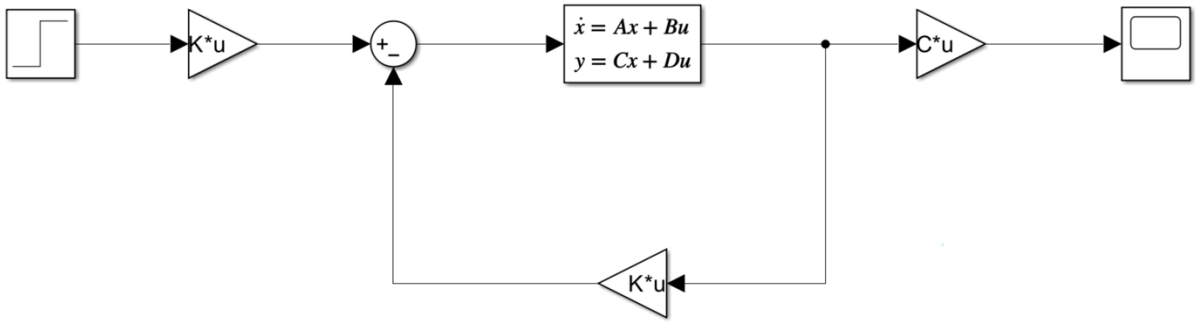

Figure 30 Model

Put $\widehat{u_d} = \begin{bmatrix}21.4203\\31.2958\end{bmatrix}$ input this system, the output is Figure 31, the three outputs are about 1.2, 2.05, and 2.5. And they are around desired values: 1, 2, and 3, But it is obviously that there are errors between actual values and ideal y_{sp} at steady state.

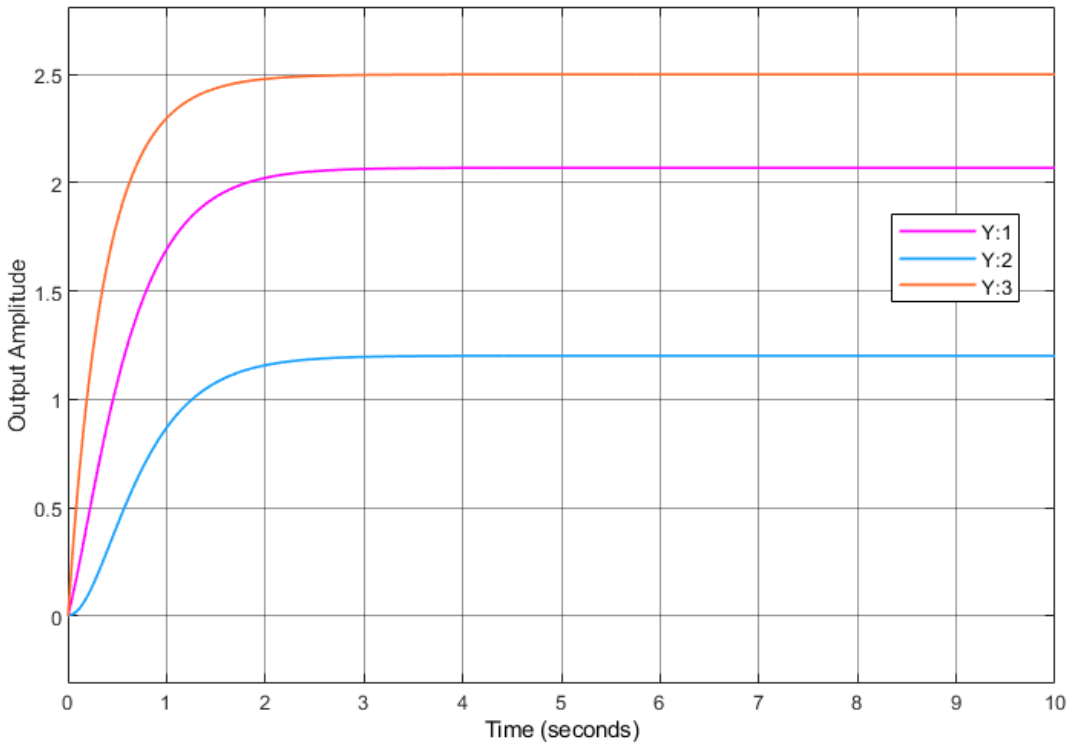

Figure 31Output of ud

8. Conclusions

In Pole-placement Control Part, after several experiments, it shows that the if the poles are further from the origin, the faster speed converging to stable state and less energy cost. I finally design the desired poles as $-2, -3, -4, -5, -6, -7$, and use unity rank method to make the closed-loop system stable, and satisfies all the requirement, and don't cost much energy.

In LQR Control part, it can be concluded that the larger the elements of Q are, the faster the state variables converge to zero with larger control signal and larger energy cost, and on the other hand, the larger the elements of R, the slower response but smaller control signal size needed with smaller energy cost. And finally I choose $Q = diag(1, 10, 20, 1, 2, 3)\ and\ R = diag(0.1, 0.1)$ to satisfy all the requirements.

Observers are always used when the sensor are expensive, and we use observers to estimate states. Comparing the LQR controller performance with observer and without observer, it is clear that at the begin, there is error between true states and estimated states, but with the closed control of observer, the error converges to zero fast and there in no difference on state and output between with and without observer in steady state.

Decoupling Control can make system easier because inputs will not influence each other on outputs. But after experiments, it is concluded that the decoupled system is BIBO stable, but not internally stable, the outputs of decoupled system can converge to stable state, but some states internally will go to infinity.

By using servo mechanism, integral control help us to do reference track and eliminate the step disturbance by adding an integrator and using augmented system.

Finally, I prove that we can't maintain the three outputs at an arbitrary constant set point with zero steady-state error because the numbers of outputs and inputs are not equal, though we can use Least-Square Estimate method to estimate the required inputs for desired outputs y_{sp}, there are still error between the actual outputs and y_{sp} at steady state.

9. Reference

[1] W. Ho, EE5103 Computer Control.

Appendix

ME5401.m

```
% initialize parameters
a = 3;
b = 1;
c = 1;
d = 9;
g = 9.8;
M_f = 2.14 + c / 20;
M_r = 5.91 - b/10;
M_c = 1.74;
L_Ff = 0.05;
L_r = 0.128;
L_c = 0.259;
J_x = 0.5+(c-d)/100;
alpha = 15.5-a/3+b/2;
gamma = 11.5+(a-c)/(b+d+3);
H_f = 0.18;
H_r = 0.161;
H_c = 0.098;
L_F = 0.133;
L_R = 0.308+(a-d)/100;
mu_x = 3.33-b/20+a*c/60;
beta = 27.5-d/2;
delta = 60+(a-b)*c/10;
den = M_f*H_f*H_f+M_r*H_r*H_r+M_c*H_c*H_c+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];
% 1:
K_1 = unity_place(A,B,[-1 -2 -3 -4 -5 -6]);
% 2:
```

```
Q = diag([1 10 20 1 2 3]);
R = 0.1*eye(2);
K_2 = my_lqr(A,B,Q,R);
% 3:
% observability matrix:
O = [C; C*A; C*A^2; C*A^3; C*A^4; C*A^5];
L = place_pole(A',C',[-5 -10 -15 -20 -25 -30])';
% 4:
C_2 = [1 0 0 0 0 0; 0 0 1 0 0 0];
% G(s) is the plant transfer function:
syms s;
G_4 = C_2 * (s * eye(6) - A) ^ (-1) * B;
B_star = [C_2(1,:)*A*B; C_2(2,:)*A*B];
C_star = [C_2(1,:)*(A+2*eye(6))^2; C_2(2,:)*(A+4*eye(6))^2];
K_4 = B_star^(-1) * C_star;
F = B_star^(-1);
s=1;
H_4 = C_2/(s*eye(6) - (A - B*K_4))*B*F;
% 5:
% initialize parameters
a = 3;
b = 1;
c = 1;
d = 9;
Y_sp = -0.1*C*A^(-1)*B*[-0.5 + (a-b)/20; 0.1+(b-c)/(a+d+10)];
% when x is stable and x_dot equals zero, we can get this Y_sp set point.
K_5 = sym('k', [2 9]);
A_5 = [A zeros(6,3); -C zeros(3,3)];
B_5 = [B;zeros(3,2)];
A_5_cl = A_5 - B_5*K_5;
Q=diag([10 9 8 7 6 5 4 3 2]);
Q=3*eye(9);
R=diag([0.1 0.1]);
K_5 = real(my_lqr(A_5, B_5, Q, R));
%6:
% numerical example:
y_sp=[1;2;3];
u = sym('u',[2 1]);
right_equ = -C*(A-B*K_2)^(-1)*B*u;
phi=[-0.0165 0.0774; -0.0580 0.0781; 0.0572 0.0407];
(phi'*phi)^(-1)*phi'*y_sp;
```

place_pole.m

```
% use cannocial form and pole placement method to place poles
% guide by nus ME5401 Chapter 7 Multiply input pole placement
% Author---Jia Yansong---2022.11.6-----
function K = place_pole(A, B, P)
    % this function is used to place poles by pole placement
    % write this for nus ME5401 mini project's six order system.
    % first three poles in P belong to first third order system,
    % second three poles in P belong to second third order system.
    % use controllable canonical form to calculate poles.
    [num_state, num_input] = size(B);
    syms s;
    %W_c = [B A*B A^2*B A^3*B A^4*B A^5*B];
    %[row, col] = size(W_c);
    index = 1;
    W_c = [];
    for i = 0: num_state - 1
        middle = A^i*B;
        W_c = [W_c middle];
        index = index + 1;
    end
    [row, col] = size(W_c);
    r = min(row, col);
    if rank(W_c) < r
        error('it is uncontrollable!');
    end
    % select num_state independent vectors out of num_state*num_input
    % vectors from the controllability matrix in the strict order form left
    % to right:
    d = zeros(1, num_input); % di imply the number of vectors in C related to the ith input, ui.
    CC = W_c(:, 1);
    j = 1;
    d(1,1) = 1;
    for i = 2: num_state*num_input
        j = j + 1;
        CC(:, j) = W_c(:, i);
        % judge this vector whether independent.
        if rank(CC) < j
            CC(:, j) = [];
            j = j - 1;
            continue;
        end
```

```
        % update d matrix in order to help next step to group them
        index = rem(i, num_input);
        if index == 0
            d(1, num_input) = i / num_input;
        else
            d(1, rem(i, num_input)) = ceil(i / num_input);
        end
        if j == num_state
            break;
        end
    end
    % group them in a square matrix:
    C = CC;
    index = 1;
    for i = 1: num_input
        for j = 0: d(1, i)-1
            C(:, index) = A^j*B(:, i);
            index = index + 1;
            if index > num_state
                break;
            end
        end
    end
    % inverse C:
    C_inv = C^(-1);
    % form T according the process:
    T = zeros(num_state, num_state);
    index = 1;
    d_index = 0;
    for i = 1: num_input
        d_index = d_index + d(1, i);
        for j = 0: d(1, i)-1
            T(index, :) = C_inv(d_index, :)*A^j;
            index = index + 1;
        end
    end
    % calculate A_bar and B_bar
    A_bar = T*A/T;
    B_bar = T*B;
    K_bar = sym('k', [num_input num_state]);
    closed_loop_matrix= A_bar - B_bar*K_bar;
```

```
    % desired close loop matrix in canonical form create num_input sub-canonical form in desired close loop
matrix:
    % create this matrix from top left to right down:
    A_cl = zeros(num_state, num_state);
    d_index = 0;
    for i = 1: num_input
        pre_index = d_index + 1;
        d_index = d_index + d(1,i);
        sub_poles = P(1, pre_index : d_index);
        sub_co = 1;
        [~, num_po] = size(sub_poles);
        for j = 1: num_po
            sub_co = sub_co * (s - sub_poles(1, j));
        end
        co = expand(sub_co);
        co1 = sym2poly(co);
        canonical_co = -1*flip(co1);
        caco = canonical_co(1,1:num_state/num_input);
        sub_matrix = create_canonical(caco);
        A_cl((i-1)*num_state/num_input + 1 : i*num_state/num_input, (i-1)*num_state/num_input + 1 :
i*num_state/num_input) = sub_matrix;
    end
    A_cl;
    d;
    % solve k
    d_index = 0;
    for i = 1: num_input
        d_index = d_index + d(1,i);
        eqn(i, :) = closed_loop_matrix(d_index, :) == A_cl(d_index, :);
    end
    sol = solve(eqn, K_bar);
    sol = struct2cell(sol);
    index = 1;
    for i = 1: num_state
        for j = 1: num_input
            K_bar(j, i) = vpa(str2num(string(sol(index))));

            index = index + 1;
        end
    end
    K = double(K_bar*T);
end
```

```
function matrix = create_canonical(co)
% co is a matrix contains non-trail row of desried canonical form matrix
    [~, num] = size(co);
    if num == 1
        matrix = co;
    else
        right_top = eye(num - 1);
        top = [zeros(num-1, 1) right_top];
        whole = [top; co];
        matrix = whole;
    end
end
```

my_lqr.m

```
function K = my_lqr(A, B, Q, R)
    % solve ARE:
    % find the positive definite solution of the Riccati equation
    syms P;
    eqn_ARE = P*A + transpose(A)*P - P*B*inv(R)*transpose(B)*P + Q == 0;
    tau = [A -B*inv(R)*transpose(B); -Q -transpose(A)];
    % get eigenvalues
    [V, D] = eig(tau);
    [row, col] = size(D);
    % store stable eignvalues' index
    index = 1;
    for i = 1: row
        if D(i,i) < 0
            d(1,index) = i;
            index = index + 1;
        end
    end
    [row_ele, col_ele] = size(d);
    index = 1;
    for i = 1: col_ele
        ele(:, index) = V(:, d(1, i));
        index = index + 1;
    end
    [r, c] = size(ele);
    v = ele(1:0.5*r, :);
    mu = ele(0.5*r+1:r, :);
    P = mu/v;
    K = real(inv(R)*transpose(B)*P);
end
```

unity_place.m

```
% place use Ackermann's formula unity rank method
% Author Jia Yansong
function K = unity_place(A, B, P)
    [num_state, num_input] = size(B);
    syms s;
    q = create_q(num_input);
    B = B*q;
    W_c = [B A*B A^2*B A^3*B A^4*B A^5*B];
    [row, col] = size(W_c);
    r = min(row, col);
    if rank(W_c) < r
        error('it is uncontrollable!');
    end
    % select num_state independent vectors out of num_state*num_input
    % vectors from the controllability matrix in the strict order form left
    % to right:
    C = W_c;
    % inverse C:
    C_inv = C^(-1);
    co = 1;
    for i = 1: num_state
        co = co * (s - P(1,i));
    end
    co = expand(co);
    coco = sym2poly(co);
    phid = polyvalm(coco, A);
    K = create_one(num_state)*C_inv*phid;
    K = q*K;
end
function res = create_q(num_input)
    res = zeros(num_input, 1);
    res(num_input, 1) = 1;
    res = [0.1;1];
end
function res = create_one(num_state)
    res = zeros(1, num_state);
    res(1, num_state) = 1;
end
```