National University of Singapore

College of Design and Engineering

Study on controller design for self-balancing vehicles

Author:
Yihui Chen

Supervisor:
Xiang Cheng

Assignment for EE5101/ME5401 Linear System

Matriculation Number: A0263115N

Email Address: e1010473@u.nus.edu

November 7, 2022

Abstract

Self-balancing two-wheeled vehicles have drawn great interest for their high safety and low energy cost. State space models can be established according to the structure for controller design. In this report, state feedback controllers are designed to stabilize the system by pole placement and LQR method. Performance of different methods are compared by simulation results. To monitor state variables, an observer is also designed followed by a discussion on effects of observer poles. Then we investigate the state feedback decoupling controller on a 2-input-2-output system. Finally a servo controller is designed to track the given reference signal regardless of disturbance. The multivariable integral controller for this system is proved to only achieve minimum steady-state error but unable to eliminate it.

Keywords: Self-balancing vehicles; Pole placement; LQR control; Observer; Decoupling; Integral controller

Contents

1 Introduction

Combining the high safety of car and low cost of motor, self-sustaining two-wheeled vehicle (Fig. 1) has drawn research interest in universities. Though most of study are still in experimental stage, different control methods have been tested on this mortorcycle-like vehicle and some exprimental results have been released online.

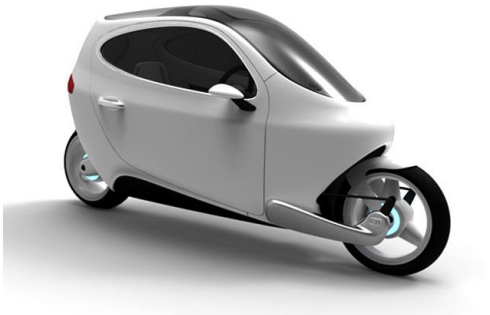

Figure 1: Two-wheeled self-balancing car

The two-wheeled vehicle is composed of a cart system, a steering system (front part) and a body (rear part). Driven by the DC servo motor, the cart and steering systems allow drivers to change cart position and handle angle. To simplify the modeling in this mini-project, the self-balance two-wheeled vehicle is assumed to be stationary and the mechanical structure is given in Fig. 2.

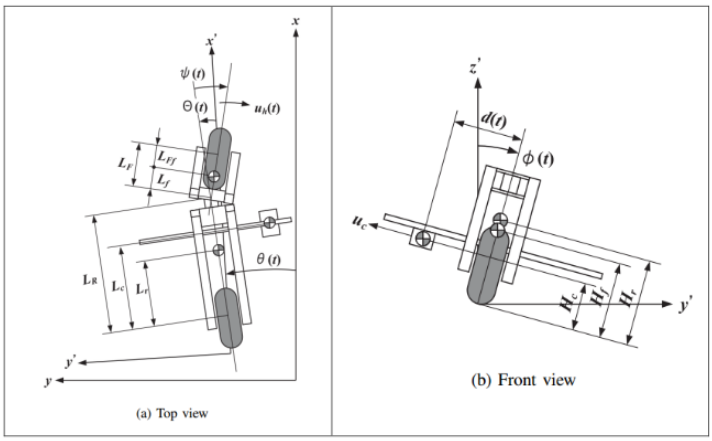

Figure 2: Simple two-wheeled vehicle model

Then, state space model is established

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

where the state is designed according to the cart position, handle angle, bike angle and their corresponding velocities respectively

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

and the relative matrices and input vectors are

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 6.5 & -10 & -\alpha & 0 & 0 \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ 5 & -3.6 & 0 & 0 & 0 & -\gamma \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \beta & 11.2 \\ b_{51} & b_{52} \\ 40 & \delta \end{bmatrix} \tag{3}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} u_c(t) & u_h(t) \end{bmatrix}^T \tag{4}$$

The parameters in matrix A and B can be calculated as

$$\begin{aligned} &a_{51} = -\frac{M_c g}{den}, a_{52} = \frac{(M_f H_f + M_r H_r + M_c H_c) g}{den}, a_{53} = \frac{(M_r L_r L_F + M_c L_c L_F + M_f L_{Ff} L_R) g}{(L_R + L_F) den} \\ &a_{54} = -\frac{M_c H_c \alpha}{den}, a_{55} = -\frac{\mu_x}{den}, a_{56} = \frac{M_f H_f L_{Ff} \gamma}{den} \\ &b_{51} = \frac{M_c H_c \alpha}{den}, b_{52} = -\frac{M_f H_f L_{Ff} \delta}{den}, den = M_f H_f^2 + M_r H_r^2 + M_c H_c^2 + J_x \end{aligned} \tag{5}$$

The physical parameters appering in Eq. 4 and Eq. 5 are usually measured mannually by experiments and they are given in appendix.

After obtaining the linear system model and giving a step reference signal for each input channel, two basic response performance specifications are required to meet as follows

- The overshoot of system should below 10%.
- The 2% settling time should be less than 5 seconds.

The rest of the report is structured in the following manner: In section 2 a state feedback controller is designed using pole placement methods, followed by a discussion on effects of different pole choices. Section 3 concludes a Linear Quadratic Regulator controller design as well as the techique on choosing the weighting matrixes Q and R. Section 4 introduces a state observer based on the former LQR control system and its performance is evaluated by monitoring the state estimation error. Section 5 describes a decoupling controller for a 2-input-2-output system. Section 6 designes a controller enabling the output of plant to track the reference signal regardless of step disturbances in input channel. Section 7 tries to track an arbitrary constatnt set point using the multivariable integral control and PID controller. Section 8 concludes the work.

2 State feedback controller by pole placement method

2.1 Method description

Given the acquired six-order state space model, feedback controller is required to make the system stable. In this section, pole placement method will be used to obtain the feedback gain to meet control requirements. Firstly considering a second-order system, its overshoot and 2% settling time of a step response is given as follows

$$M_p = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}, t_s = \frac{4.0}{\zeta\omega_n} \tag{6}$$

where ζ and ω_n are damping ratio and natural frequency respectively. Now that the overshoot should below 10% and settling time less than 5s, we set $\zeta = 0.8$ and $\zeta\omega_n = 2.5$ in this case, and then the two pole of second-order system is calculated as

$$\lambda_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{7}$$

To assure the stability of system, the two dominant poles should have little smaller real parts, so we set them at $\lambda_{1,2} = -2.5 \pm j1.875$. The other 4 poles are chosen 4-5 times lefter than the two dominant poles. Then the desired characteristic polynomial becomes

$$\begin{aligned}\phi_d(s) &= (s-\lambda_1)(s-\lambda_2)(s-\lambda_3)(s-\lambda_4)(s-\lambda_5)(s-\lambda_6) \\ &= s^6 + a_5s^5 + a_4s^4 + a_3s^3 + a_2s^2 + a_1s + a_0\end{aligned} \tag{8}$$

The desired closed-loop matrix can be written in controllable canonical form

$$A_d = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -a_5 & -a_4 & -a_3 & -a_2 & -a_1 & -a_0 \end{bmatrix} \tag{9}$$

Then the remainning task is transform the system with state feedback also to the controllable canonical form. First compute the controllability matrix and check if it is full rank

$$W_c = \begin{Bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B \end{Bmatrix} \tag{10}$$

Then select 6 independent vectors from W_c in the strict order from left to right (the first six column vectors in this case) and group them in matrix X in the following form

$$X = \begin{Bmatrix} b_1 & b_2 & Ab_1 & Ab_2 & A^2b_1 & A^2b_2 \end{Bmatrix} \tag{11}$$

Also compute the inverse of X and write it as the following form

$$X^{-1} = \begin{bmatrix} q_1^T & q_2^T & q_3^T & q_4^T & q_5^T & q_6^T \end{bmatrix}^T \tag{12}$$

Because that the system has multiple input channels, choose the $3^{th}(d_1 = 3)$ and $6^{th}(d_1 + d_2 = 6)$ row vectors of X^{-1} and construct transformation matrix T as

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

Define the feedback gain matrix $\bar{K} \in \mathbb{R}^{2\times6}$ with 12 unknown variables, we can solve it by establishing the equation

$$\bar{A} - \bar{B}\bar{K} = A_d \tag{14}$$

where $\bar{A} = TAT^{-1}$, $\bar{B} = TB$

After obtainning the matrix $\bar{K}$, we can finally get the original feedback gain matrix $K = \bar{K}T$. Substitute the control law $u = -Kx + r$ into Eq. 1 and the new closed-loop system becomes

$$\begin{aligned}\dot{x} &= (A - BK)x + Br \\ y &= Cx\end{aligned} \tag{15}$$

2.2 Simulation results and refinements

In this report, the 6 poles are selected firstly as $\lambda_{1,2} = -2.5 \pm j1.875, \lambda_3 = -10, \lambda_4 = -10.75, \lambda_5 = -11.5$ and $\lambda_6 = -12.5$. The final feedback gain matrix K is calculated to be

$$K = \begin{bmatrix} -15.68 & 12.20 & 6.27 & 4.90 & -6.21 & -2.26 \\ 250960 & -21110 & -107660 & -845.58 & 9043.2 & 412.98 \end{bmatrix}$$

The number of elements differes a lot and are relatively weird. Temporarily we use the feedback matrix and simulate the closed-loop system with time domain from 0 to 10 seconds. Given a step signal for each input channel, the output response is showed in Fig. 3. Sadly though the system becomes stable with a short settling time, there are huge overshoots in both cases and the stable points are also impossible in real cases.

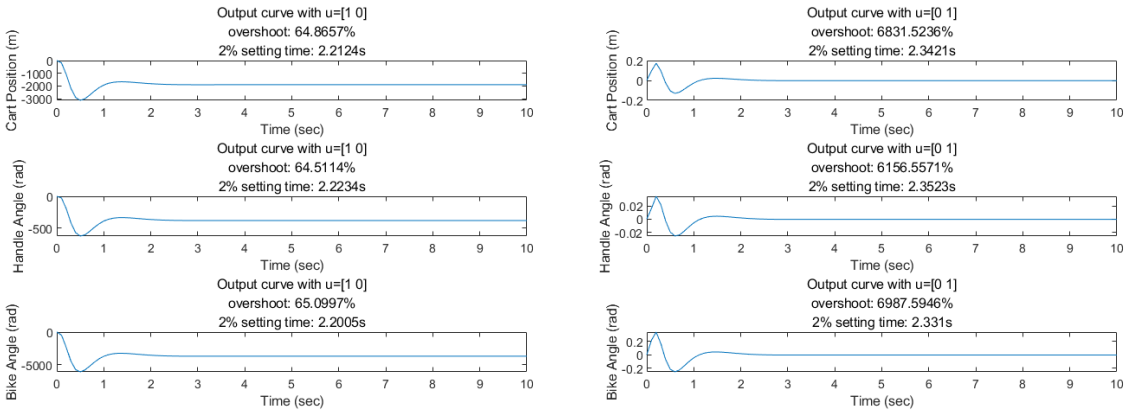

Figure 3: Step response for the closed-loop system

The results are unacceptable and need further refinements. To improve the performance, we use another A_d with the same eigenvalues but not in controllable canonical form.

$$A_d = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -a_5 & -a_4 & -a_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -a_2 & -a_1 & -a_0 \end{bmatrix}$$

In this case, we remain all poles unchanged. The resultant feedback gain matrix K' is given as below and the simulation results are shown in Fig. 4. Though large elements exist in K', the stable point of both cases are reasonable and the overshoots are greatly reduced, enabling the system to meet the given basic design specifications.

$$K' = \begin{bmatrix} 250820 & -210790 & -107620 & -846.8122 & 9053.3 & 411.9200 \\ 128.1535 & -305.4679 & -31.6386 & 6.1301 & -16.3902 & -1.2010 \end{bmatrix}$$

We also assume that all six states can be observed and the initial condition is x_0, their response to zero external inputs is given in Fig. 5. It can been seen all states converges to zero from the non-zero initial values in less than 5 seconds and the system becomes stable.

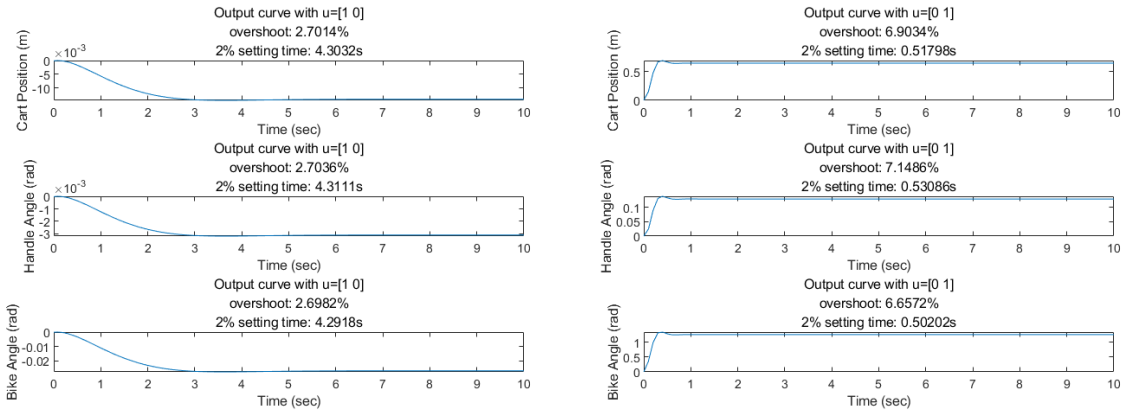

Figure 4: Step response for the closed-loop system with refined K'

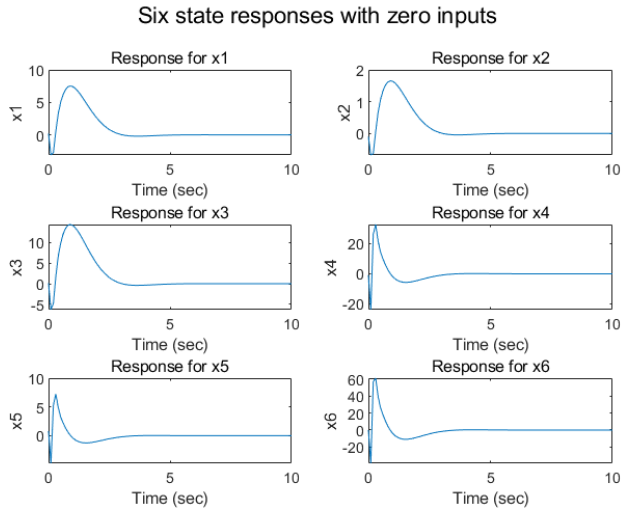

Figure 5: All state responses with zero inputs for the refined system

2.3 Comparison with "place" function in MATLAB

Actuallly when tackling pole placement problems for high order system, some pole choices will result in a large gain as in our former methods, which makes it sensitive to noise and causes numerically instable problems. One robust algorithm was proposed in [1] and has been applied in matlab function (called 'place'). The aim of this algorithm is to find the feedback gain matrix K that makes $A - BK$ the most robust. One method to measure the robustness of a matrix is to use condition number

$$c_j = \frac{1}{s_j} = \frac{\|y_j\|_2 \|x_j\|_2}{\left|y_j^T x_j\right|} \tag{16}$$

where x_j and y_j are the left and right eigenvalues of $A-BK$ respectively. It can be proved that c_j has a upper bound i.e. $max_j c_j \leq \kappa_2(X) \equiv \|X\|_2 \left\|X^{-1}\right\|_2$, where $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ is the eigenvector matrix. The maximum value $v = max(c_j)$ of all condition numbers is chosen and the system is said to be the most robust if v is the smallest. Then the problem becomes how to find the minimum upper bound of condition numbers.

The minimum upper bound can ben found by cauchy inequality and the result is given as follows

$$min(\kappa_2(X)) \leq \frac{\kappa_2(S)}{\sqrt{k}} \tag{17}$$

S is orthogonal normal basis of the null space of $N(U_1^T(A-\lambda I))$, where U_1 is the orthogonal basis of null space of input matrix B, λ are eigenvalues of matrix $A-BK$ and k is the dimensions of S. QR decomposition or singular value decomposition (SVD) can be used to calculate the orthogonal basis. Finally, feedback gain matrix K can be obtained by

$$K = Z^{-1}U_0^T(A - X\Lambda X^{-1}) \tag{18}$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$, Z is the Q matrix of B and U_0 is also orthogonal basis of B.

The same poles are chosen but we use the new method to calculate feedback matrix K'', the result is given below and we can see the gains become smaller.

$$K'' = \begin{bmatrix} 8.7165 & -3.6746 & -1.8207 & 0.5685 & -0.4985 & -0.0880 \\ -11.2971 & 10.6474 & 5.4543 & -0.8739 & 1.4368 & 0.3794 \end{bmatrix}$$

The step response for the new closed-loop system is shown in Fig. 6, where both the overshoot and settling time meet the requirements. Compared with our former results, this system is stabilized more quickly and overshoots are smaller in most cases. All six state responses with non-zero initial condition x_0 and zero external inputs is given in Fig. 7. Compared to our refined method with gain matrix K', it achieves much smaller overshoots as well as less settling time. Moreover, the robust system matrix given by "place" function in matlab will also be less sensitive to noise than our simple method.

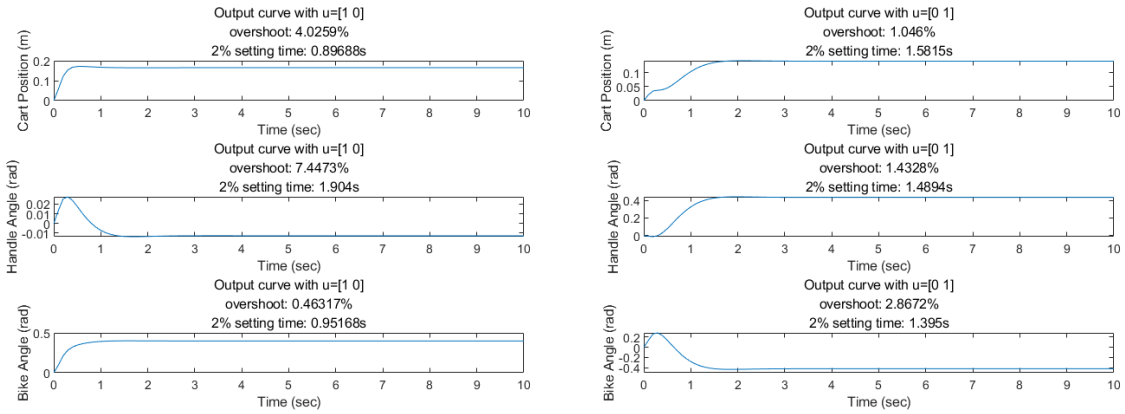

Figure 6: Step response for the new closed-loop system by "place" function

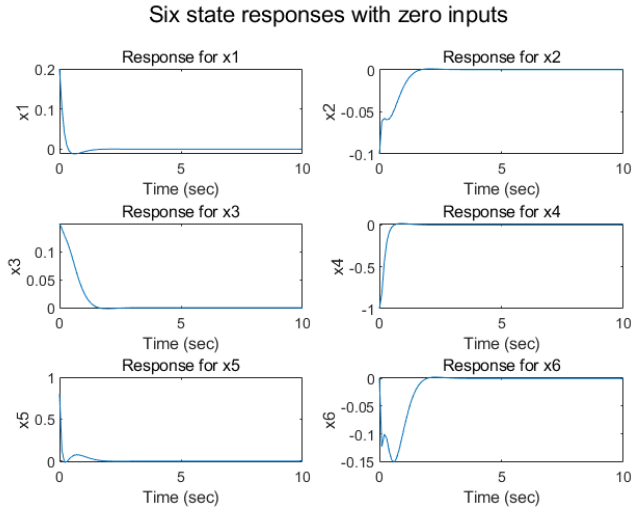

Figure 7: All state responses with zero inputs by "place" function

2.4 Discussion on pole positions

Here we still choose our self-designed feedback gain matrix K' to study effects of different poles on system performance. Although one can always stabilize the system and make response faster by designing poles on the very left plain, the input cost will increase as well. To investigate the influence of poles, two more groups of poles are used. The new poles are listed as following.

$$p' = \begin{bmatrix} -1.25 + j1.875 & -1.25 - j1.875 & -5 & -5.375 & -5.75 & -6.25 \end{bmatrix}$$
$$p'' = \begin{bmatrix} -5 + j1.875 & -5 - j1.875 & -20 & -21.5 & -23 & -25 \end{bmatrix}$$

According to the new poles, two closed-loop system are established. Fig. 8 gives a comparison between their zero input response with non-zero initial state x_0. It is clear that the system response will be faster if designed poles have larger negative real parts. On the other side, Fig. 9 illustrates that faster poles also result in a higher control cost.

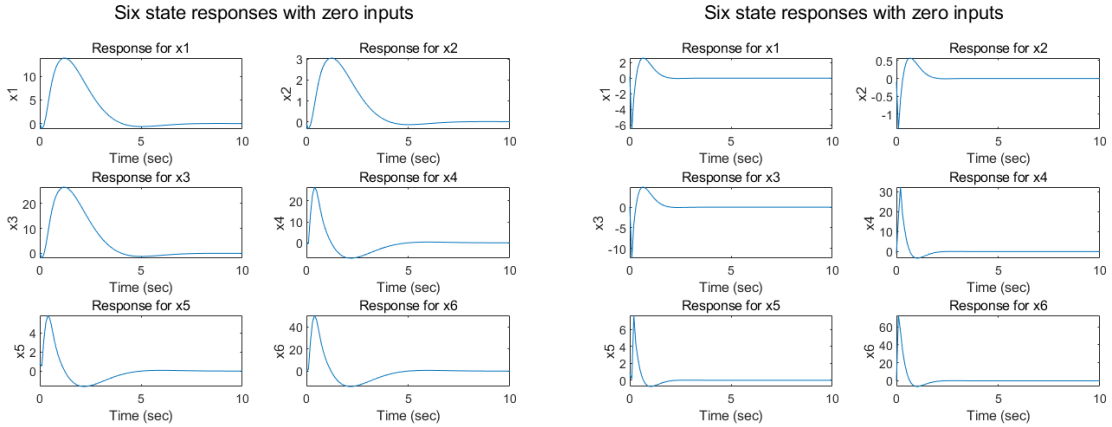

Figure 8: Step response and control cost of system with poles p

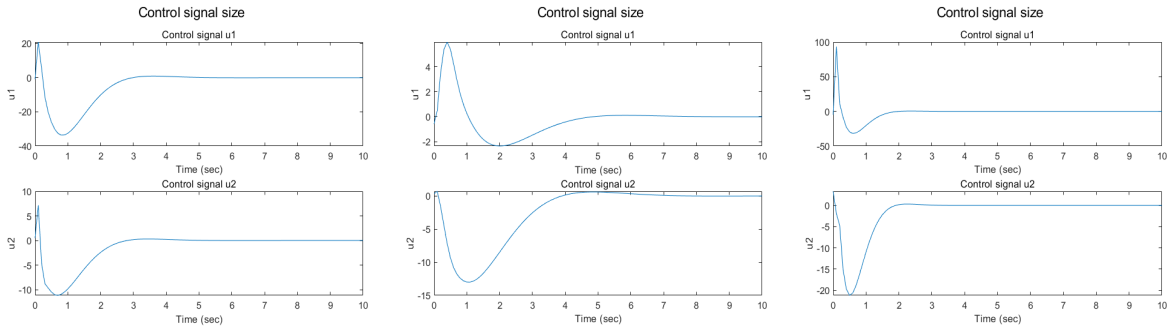

Figure 9: Control signal size with p, p' and p'' respectively

Pole placement methods have the magic of easily stabilizing the system and ensuring the expected design performance by using state feedback controller. On the other hand, poles of systems are hard to select and have no strict standards. In next section, linear quadratic regulator will be introduced to solve the trade-off problem between system performance and control cost.

3 Linear quadratic regulator controller

3.1 Method description

3.1.1 Class method

Linear quadratic resgulator (LQR) follows the theory of optimal control, which is aimed to minimized the cost function. Given a multi-input-multi-output system and consider both control performance and input cost, the cost function can be defined as

$$J = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u) dt \tag{19}$$

The Q and R in Eq. 19 are weighting matrixes representing cost on response speed and energy efficiency respectively. After selecting the appropriate weighting matrix, a state feedback controller $u = -Kx + r$ can be designed to minimize the cost function, where the gain matrix $K = R^{-1}B^T P$ and matrix P is given by solving the Riccati equation

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \tag{20}$$

Then the problem becomes solving the algebraic matrix riccati equation. One method is to use an eigenvalue-eigenvector based algorithm. First a $2n \times 2n$ matrix is derived

$$\Gamma = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{21}$$

Then n stable eigenvalues of Γ are found as well as their corresponding eigenvectors. Write the eigenvectors in the form of $\begin{pmatrix} v_i & u_i \end{pmatrix}^T, i = 1, 2, \cdots, n$. Then the P matrix in riccati equation can be calculated by

$$P = \begin{pmatrix} \mu_1 & \cdots & \mu_n \end{pmatrix} \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix}^{-1} \tag{22}$$

3.1.2 Discrete LQR method

Another method is given by considering a discrete case. Similarly, a discrete cost function is given in Eq. 23.

$$J = \frac{1}{2}\sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \tag{23}$$

The first step is to discretize the state space equation. According to the mean value theorem of integral

$$x(t + dt) = x(t) + Ax(\xi)dt, \xi \in (t, t + dt) \tag{24}$$

where dt is the dicrete time interval set mannually. Here we use midpoint euler method ($x(\xi) = \frac{x(t)+x(t+dt)}{2}$) to estimate the next state, we got

$$x(t + dt) = (I - \frac{Adt}{2})^{-1}(I + \frac{Adt}{2})x(t) \tag{25}$$

Back to the state space equation in Eq. 1, we integrate both sides and get

$$\int_t^{t+dt} \dot{x} = \int_t^{t+dt} (Ax + Bu)dt \Rightarrow x(t + dt) - x(t) = Ax(\xi)dt + Bu(\xi)dt \tag{26}$$

By substituting Eq. 25 into it and take some approximation, we finally get

$$x(t+dt) = (I - \frac{Adt}{2})^{-1}(I + \frac{Adt}{2})x(t) + Bdtu(t) \tag{27}$$

Now we can rewrite Eq. 27 into a form of dicrete state space model

$$x(k+1) = \bar{A}x(k) + \bar{B}u(k) \tag{28}$$

where $\bar{A} = (I - \frac{Adt}{2})^{-1}(I + \frac{Adt}{2})$ and $\bar{B} = Bdt$.

The second step is to solve the discrete riccati equation

$$P = Q + \bar{A}^T P\bar{A} - \bar{A}^T P\bar{B}(R + \bar{B}^T P\bar{B})^{-1}\bar{B}^T P\bar{A} \tag{29}$$

P can be solved numerically by iteration method. Usually we give P a initial value as Q and P will quickly converges after a number of iterations. With the matrix P, we finally get the feedback gain matrix $K = (R + \bar{B}^T P\bar{B})^{-1}\bar{B}^T P\bar{A}$.

3.2 Simulation results

Like what we did in Section 1, we first evaluate the closed-loop system performance under a step signal input for each channel with zero initial conditions. Since we obtain two methods for LQR controller, a comparison between them (with the same weighting matrix Q and R) is also given.

We initially select weighting matrix $Q = diag\begin{Bmatrix}10 & 50 & 50 & 5 & 50 & 50\end{Bmatrix}$ and $R = I_{6\times 6}$. With the first eigenvalue-based method, the feedback gain matrix K is acquired as

$$K = \begin{pmatrix} 54.3246 & -45.5065 & -19.2405 & 5.2022 & -5.3849 & 2.3616 \\ -45.7713 & 39.8256 & 25.5740 & -4.0248 & 5.4445 & 6.9241 \end{pmatrix}$$

Then the closed-loop system is simulated as is shown in Fig. 10. The system is stabilized and successfully meet the design specifications. With the selected weighting matrix Q and R, there exists no overshoot and the settling time are all around 4s.

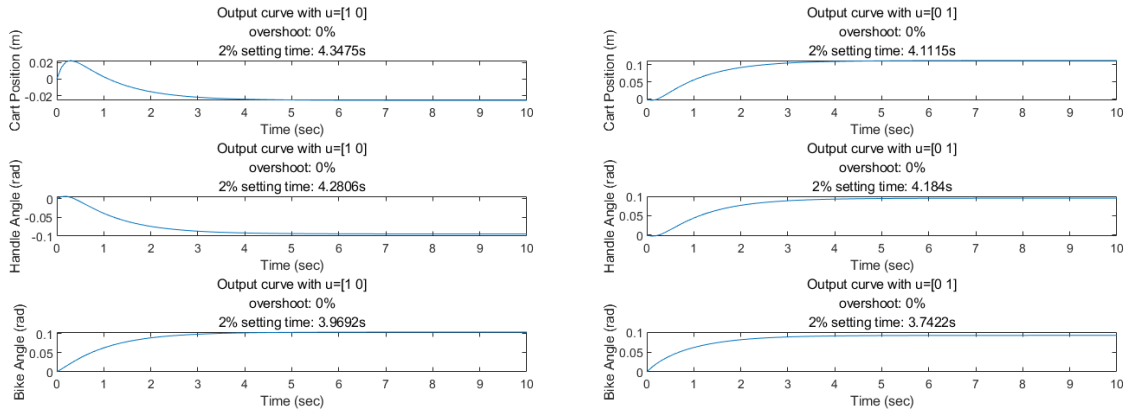

Figure 10: Step response for closed-loop system by eigenvalue-based LQR method

Then the second discrete LQR method is applied. We suppose that the matrix P converges when the mean of elements in $\mid P_{new} - P_{old} \mid$ is less than a small number. The gain matrix K is calculated as

$$K = \begin{pmatrix} 46.7765 & -39.2025 & -19.5779 & 4.2947 & -4.8494 & -0.6493 \\ -33.3206 & 28.1860 & 15.6346 & -3.0081 & 3.6233 & 1.9127 \end{pmatrix}$$

The simulation results is shown in Fig. 11. Compared with the eigenvalue-based one, this discrete lqr method have a little larger overshoot but also cut down the settling time in most cases. The dlqr method is also computationally efficient because the matrix P will quickly converges while the first method always

need to handle the eigenvalue problem. However, when step signal is added in first input channel, the settling time of response in the second output channel is over 5s and unable to meet the basic design specifications. Therefore we will still use the eigenvalue-based method in the following discussion.

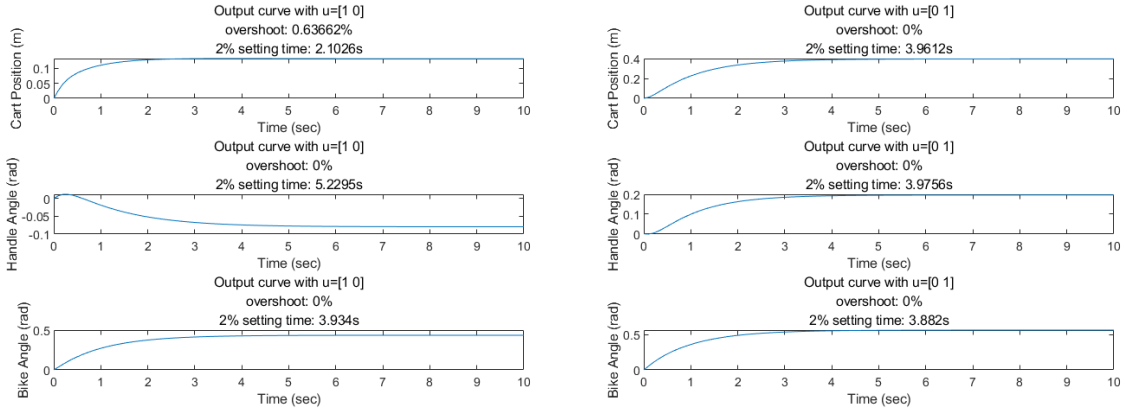

Figure 11: Step response for closed-loop system by discrete LQR method

Fig. 12 also gives the six state responses to non-zero initial state condition x_0 with zero external inputs. All states becomes stable at zero point in a short time as expected.

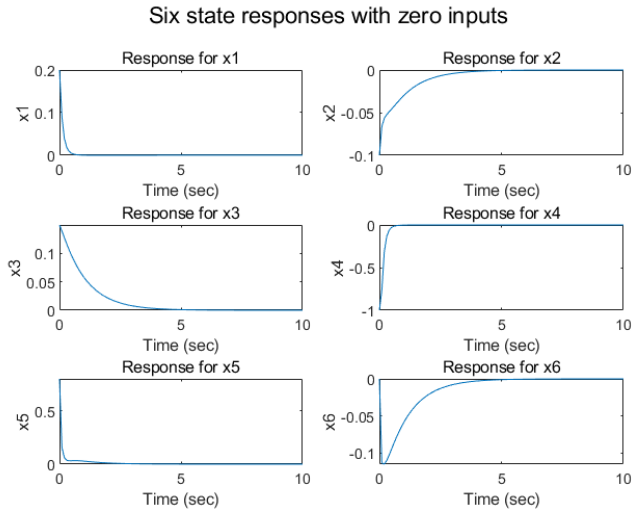

Figure 12: State response with zero external inputs

3.3 Discussion

Rather than design poles to stabilize the system, LQR method achieves the stability by choosing reasonable weighting matrix Q and R. To investigate their effects on system, we remain R unchanged and multiply the Q matrix with a scalar to form $Q' = \begin{Bmatrix} 5 & 25 & 25 & 2.5 & 25 & 25 \end{Bmatrix}$. By monitoring the control signal cost and comparing it with the original system (Fig. 13), we can find that with a smaller Q, the energy cost is reduced. We also keep Q unchanged and design a new R matrix with bigger weights. The simulation results is given in Fig. 14, where the bigger R gives a greater punishment on control energy and LQR controller automatically cut down the control signal.

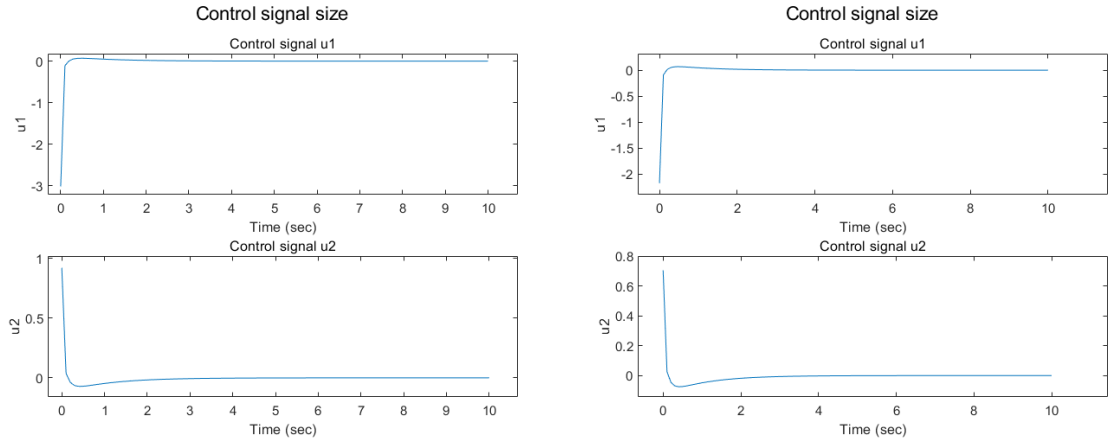

Figure 13: Comparison between control cost with Q and Q' (R remain unchanged)

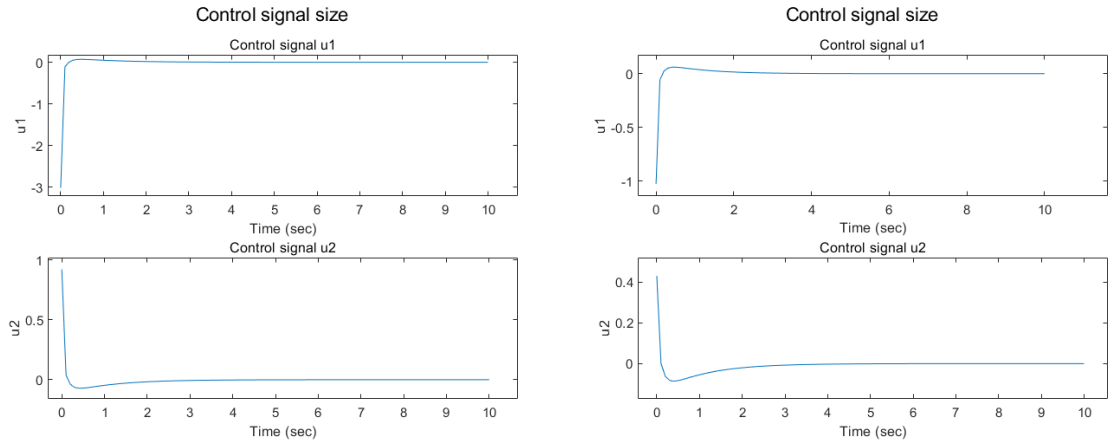

Figure 14: Comparison between control cost with R and R' (Q remain unchanged)

Different from the pole placement methods, LQR method offers a way to do trade-off between response speed and control signal cost by tuning the weighting matrixes. In practical situation, one can define the most reasonable cost function according to the established mathematic model. Unlike selecting poles by experience and check system performance repeatedly, LQR method can always find the optimal solution as long as the cost function is given.

4 Observer-based LQR control system

4.1 Method description

In practice, not all state variables can be measured for a system and an observer is required to estimate these unseen states. Suppose the estimated state is $\hat{x}(t)$ and the true state is $x(t)$, then the state estimate error is

$$\tilde{x}(t) = x(t) - \hat{x}(t) \tag{30}$$

We hope the error dynamics is adjustable and converges to zero with time, thus a closed-loop estimator is used (Fig. 15).

The state space equation for estimator is derived in Eq. 31 and also the state estimation error in Eq. 32.

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L(y - C\hat{x}) \tag{31}$$

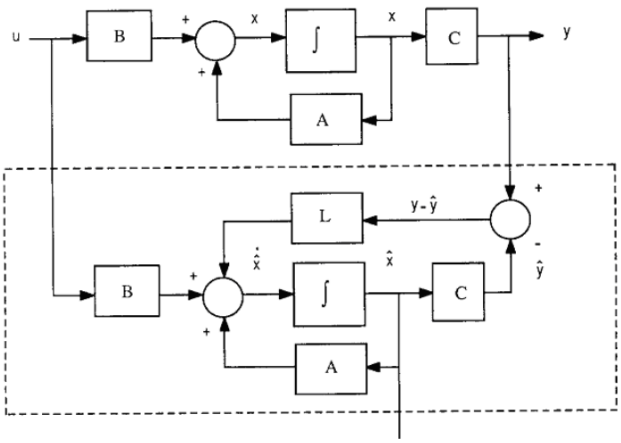

Figure 15: A closed-loop estimator

$$\dot{\tilde{x}}(t) = (A - LC)\tilde{x} \tag{32}$$

Then observer poles can be selected to get the observer gain matrix L using pole placement algorithms. Remember in early sections we simply use the invisible state variables to design the state feedback controller, which is unpractical in actual situations. Now we can implement the estimated state $\hat{x}$ in our LQR controller and evaluate the new closed-loop system. Consider the control law $u = -K\hat{x} + r$ and substitute it into the original state space equation, and also combine with Eq. 32 to form a new system

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{\tilde{x}} \end{bmatrix} &= \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} r \\ y &= Cx \end{aligned} \tag{33}$$

From Eq. 33, we can deduce the Laplace transform of the combined system

$$\begin{aligned} X(s) = (sI - A + BK)^{-1}x(0) + (sI - A + BK)^{-1}BK(sI - A + LC)^{-1}\tilde{x}(0) + (sI - A + BK)^{-1}BR(s) \\ \tilde{X}(s) = (sI - A + LC)^{-1}\tilde{x}(0) \end{aligned} \tag{34}$$

It is clear that there would be no state estimation error if the initial estimated state is the same with real one ($\tilde{x}(0) = 0$). Now that we couldn't acquire the real $x(0)$ in most cases, we assume our initial estimated $\hat{x}(0)$ to be zero.

4.2 Simulation results and discussion

First we select observer poles 2.7 times faster than the control poles p used in section 1. Also we modeify the Q and R in LQR controller to achieve a better system performance. The new weighting matrix are $Q = diag\begin{Bmatrix} 4 & 22 & 22 & 4 & 22 & 22 \end{Bmatrix}$ and $R = 5I$ respectively. The system's zero-state response with a step signal in each channel is given in Fig. 16. The six responses all have small overshoots and are stable with time, but in the left picture the settling time for the first ouput channel is a little bit longer. We can plot the state estimation error with time in Fig. 17, where all of the errors quickly converges to zero in 1s.

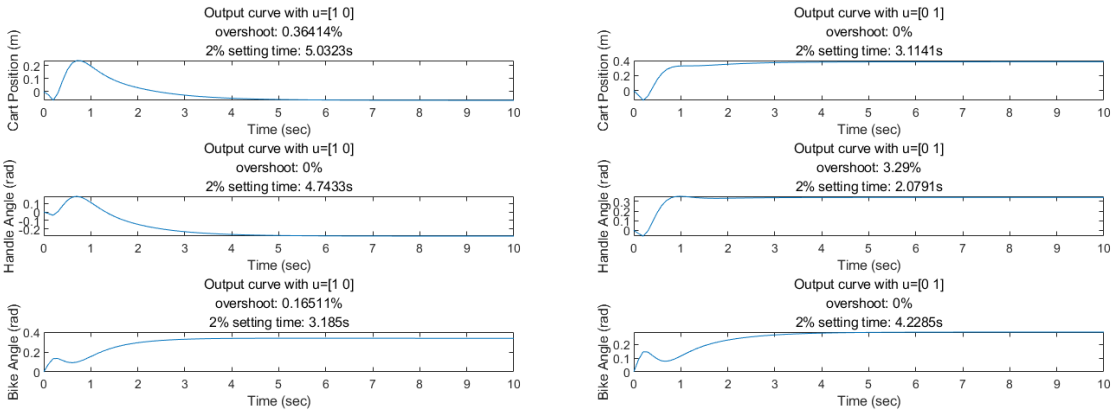

Figure 16: Step response for the observed-based system

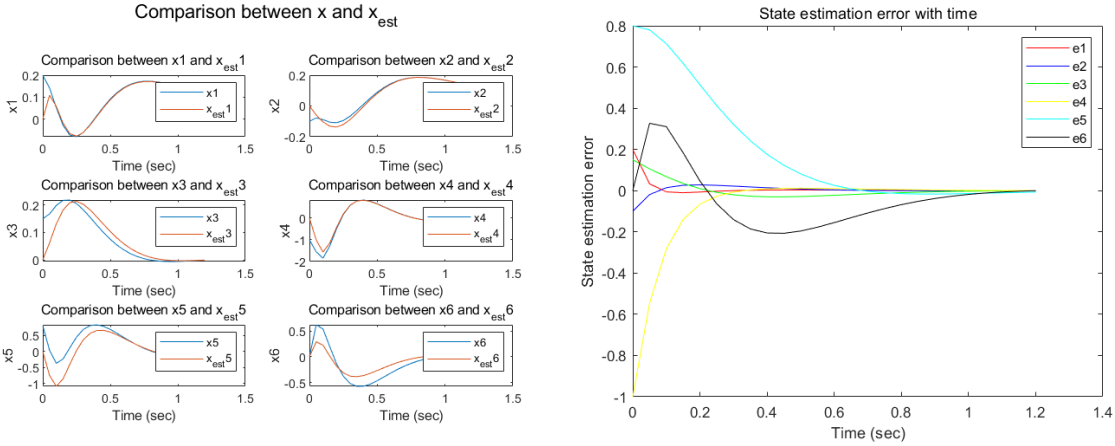

Figure 17: Investigation on state estimation error

Then we keep weighting matrixes unchanged and modify the observer poles. Two new pairs of poles are selected as 2 and 4 times faster than the control poles. The simulation results are illustrated in Fig. 18 and Fig. 19. We can see that in when the observer poles are too left, overshoots are largely increased while settling time are almost unchanged in both cases.

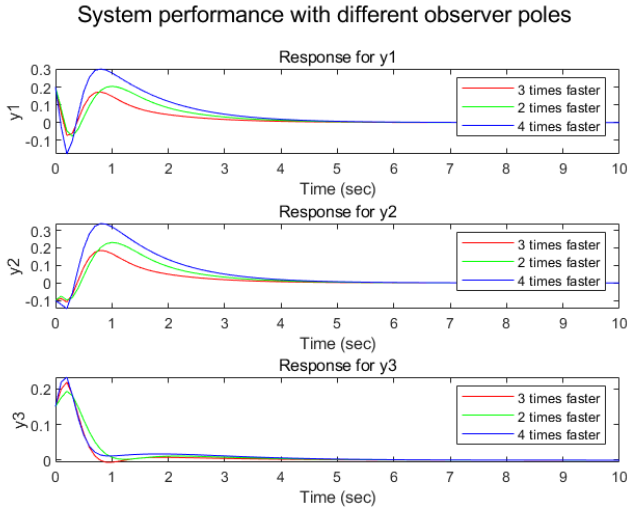

Figure 18: Comparison of system performance with different observer poles (zero input, non-zero condition)

The state estimation error with these two different pairs of poles are given in Fig. 19 and Fig. 20. It is clearly shown the state estimation error converges to zero faster with observer poles on the more left plain. However, large overshoots may also appear at the same time.

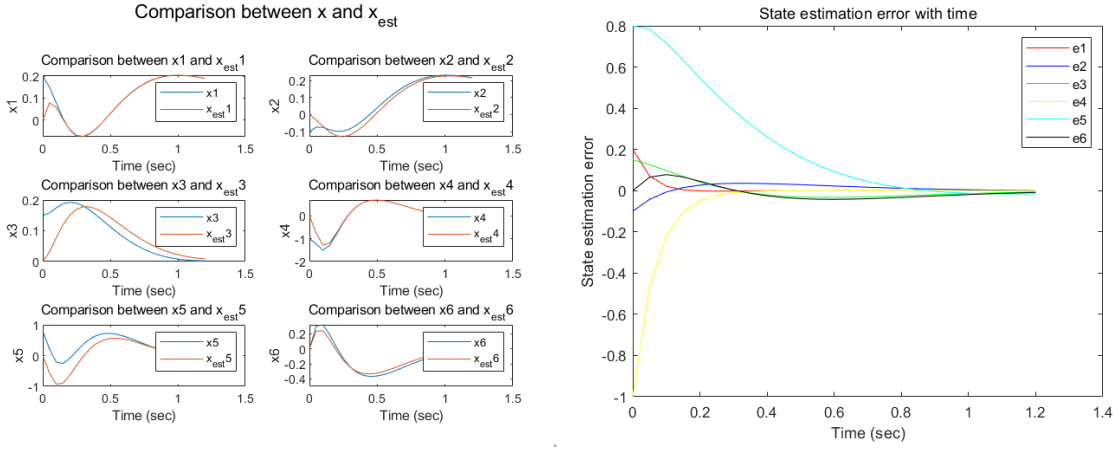

Figure 19: State estimation error with slower observer poles

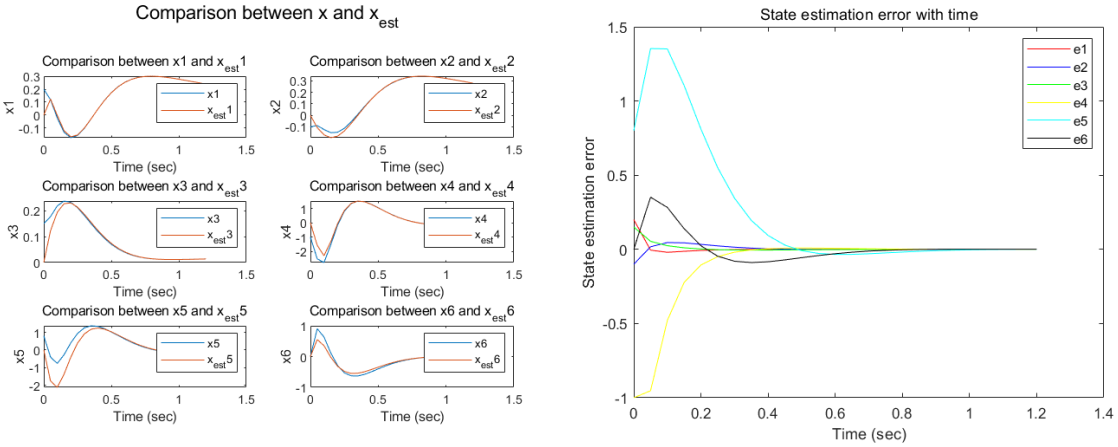

Figure 20: State estimation error with faster observer poles

5 System decoupling control

5.1 Method description

5.1.1 Decoupling by state feedback

In this section, we suppose that cart position $d(t)$ and bike angle $\psi(t)$ are the most two interesting outputs and the new output matrix is

$$C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{35}$$

Then the plant becomes a 2-input-2-output system. To achieve decoupling control, a state feedback control law $u = -Kx + Fr$ is applied such that

$$\begin{pmatrix} y_1(s) \\ y_2(s) \end{pmatrix} = \begin{pmatrix} h_{11} & 0 \\ 0 & h_{22} \end{pmatrix} \begin{pmatrix} r_1(s) \\ r_2(s) \end{pmatrix} \tag{36}$$

The closed-loop state space system becomes $\dot{x} = (A - BK)x + BFu$, and by laplace tranformation, we can relate the open-loop and closed loop transfer function matrix:

$$H(s) = G(s)[I + K(sI - A)^{-1}B]^{-1}F \tag{37}$$

where $G(s) = C(sI - A)^{-1}B$.

To decouple the system we need $H(s)$ to be diagonal and thus $G(s)$ should be non-singular. It can be proved that $G(s)$ can be written in the form of

$$G(s) = \begin{pmatrix} g_1^T(s) \\ g_2^T(s) \\ \vdots \\ g_m^T(s) \end{pmatrix} = \begin{pmatrix} s^{-\sigma_1} & 0 & \cdots & 0 \\ 0 & s^{-\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s^{-\sigma_m} \end{pmatrix} [B^* + C^*(sI - A)^{-1}B] \tag{38}$$

where $B^* = \begin{pmatrix} c_1^T A^{\sigma_1 - 1}B \\ c_2^T A^{\sigma_2 - 1}B \\ \vdots \\ c_m^T A^{\sigma_m - 1}B \end{pmatrix}, C^* = \begin{pmatrix} c_1^T A^{\sigma_1} \\ c_2^T A^{\sigma_2} \\ \vdots \\ c_m^T A^{\sigma_m} \end{pmatrix}$ and the relative degree σ_i can be derived by Eq. 39

$$\sigma_i = \begin{cases} min(j | c_i^T A^{i-1}B \neq 0^T, j = 1, 2, \cdots, n) \\ n, \quad if \quad c_i^T A^{i-1}B = 0^T \quad i = 1, 2, \cdots, n \end{cases} \tag{39}$$

Then let the matrixes in control law be $F = (B^*)^{-1}$ and $K = (B^*)^{-1}C^*$ and substitute $G(s)$ into Eq. 37, the closed-loop system transfer function matrix is $H(s) = diag(s^{-\sigma_1}, s^{-\sigma_2}, ..., s^{-\sigma_m})$, which is in a diagonal form and the system is decoupled.

Also to ensure the stability of the decoupled system, the feedback gain matrix K can be designed as

$$K = (B^*)^{-1} \begin{pmatrix} c_1^T \phi_{f1}(A) \\ c_2^T \phi_{f2}(A) \\ \vdots \\ c_m^T \phi_{fm}(A) \end{pmatrix} \tag{40}$$

where $\phi_{fi}(A) = A^{\sigma_1} + \gamma_{i1}A^{\sigma_{i-1}} + \gamma_{i\sigma_i}I$ is the stable characteristic polynomial of each input-output pair.

5.1.2 Decoupling by output feedback

Besides using state feedback methods, the system can also be decoupled by output feedback if the transfer function $G(s)$ is given. Similar with the former method, with the feedback transfer function $K(s)$, we first derive the relationship between open-loop and closed-loop transfer functions.

$$H(s) = [I + G(s)K(s)]^{-1}G(s)K(s) \tag{41}$$

It is easy to get that $H^{-1} = (GK)^{-1} + I$, which means that GK should be decoupled. In the design procedure, we first write $G(s) = \frac{N(s)}{d(s)}$, where $d(s)$ is the least common denominators of $G(s)$. Then $K_d(s) = adj(N(s))$ is chosen so that $G(s)K_d(s) = \frac{det(N(s))}{d(s)}I_m$ is successfully decoupled. Moreover, to ensure that $K(s)$ is proper and the final closed-loop system is stable, a stabilizer $K_s(s)$ should be designed using pole placement or lqr methods after getting the m decoupled single-input-single-output subsystems ($m = 2$ in this case). Finally, after the output feedback $K(s) = K_d(s)K_s(s)$ is obtained, it should be ensured with no unstable pole-zero cancellations between $G(s)$ and $K(s)$, which makes the system to be internally stable.

5.2 Simulation results and discussion

Since the state space equation has been given, we choose the state feedback method here. The relative degree σ_1 and σ_2 are checked to be both 2. Then we define the stable characteristic polynomial of each input-output pair to be $\phi_{fi}(A) = (A-\lambda_{i1})(A-\lambda_{i2}), i = 1, 2$. Here we choose λ_{11} and λ_{12} to be $-0.8 \pm j0.8$ while $\lambda_{21} = -1.6, \lambda_{22} = -1.7$. The final result of K and F in control law is given as follows.

$$K = \begin{pmatrix} 0.0197 & 0.4083 & -0.5984 & -0.7633 & 0 & 0.0892 \\ 0.0699 & -0.3311 & 0.4428 & 0.5072 & 0 & -0.1992 \end{pmatrix}, \quad F = \begin{pmatrix} 0.0570 & -0.0106 \\ -0.0378 & 0.0237 \end{pmatrix}$$

Now first check the closed-loop system's zero-state response with step signal for each input channel. From Fig. 21, it is clear that the each output channel is only influenced by its corresponding input signal, which means the system is successfully decoupled. The system also meets the basic design requirements and have a small overshoot. Fig. 22 also gives the system response with non-zero initial state x_0 and we can know the closed-loop system is external(BIBO) stable.

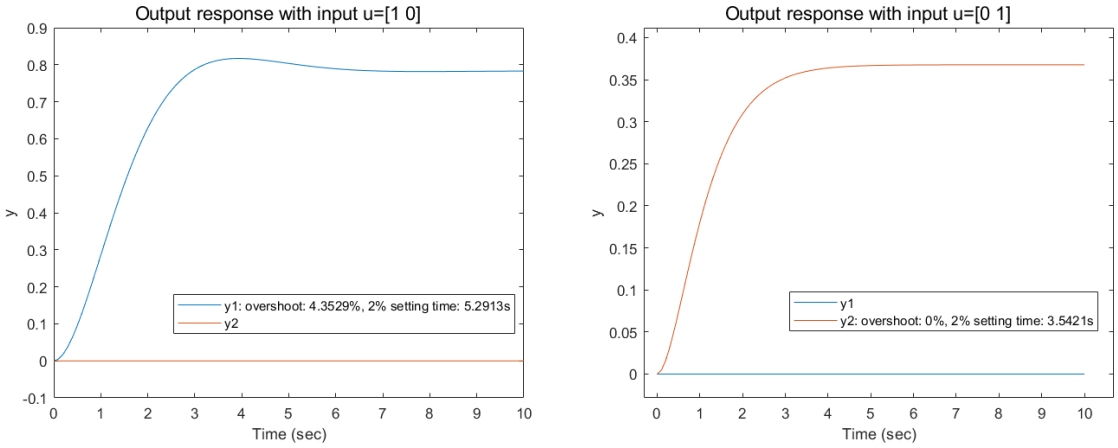

Figure 21: Step response for the decoupled system (zero initial condition)

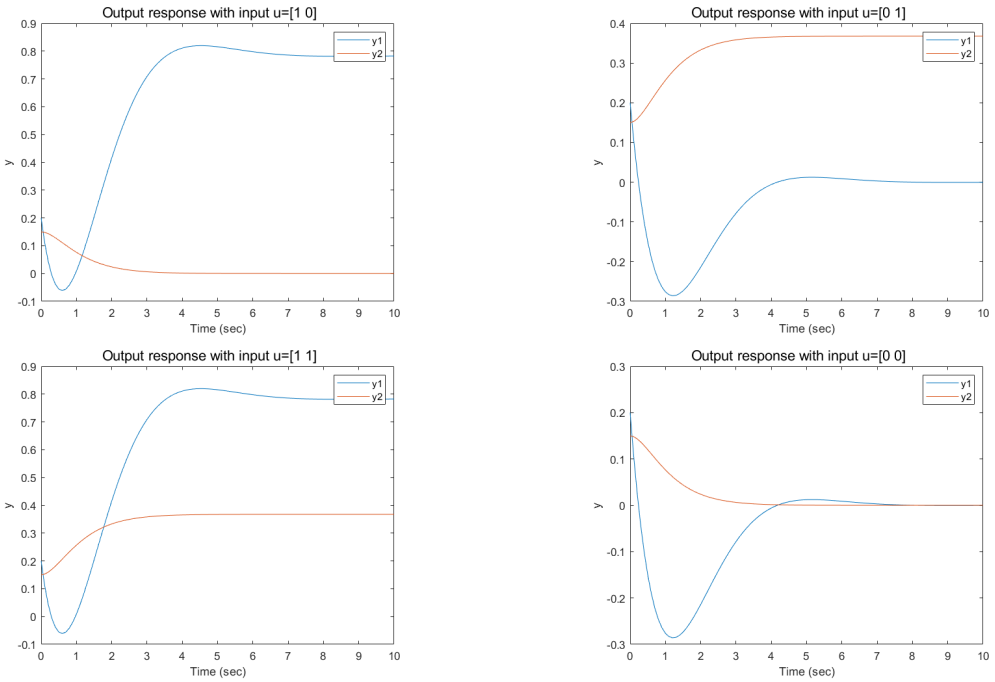

Figure 22: Step response of the decoupled system for 4 combinations of inputs (non-zero initial condition)

However, the state feedback controller can't ensure the internal stability. To investigate it, we utilize the state observer designed in Section 4 to check the six state responses after given a step signal in the first channel. The response curve is given in Fig. 23, where the second and fifth states are unstable. Therefore the system is only BIBO stable but not internally stable. The poles of the system is also checked to have one positive real part (2.4821), which results in the instability.

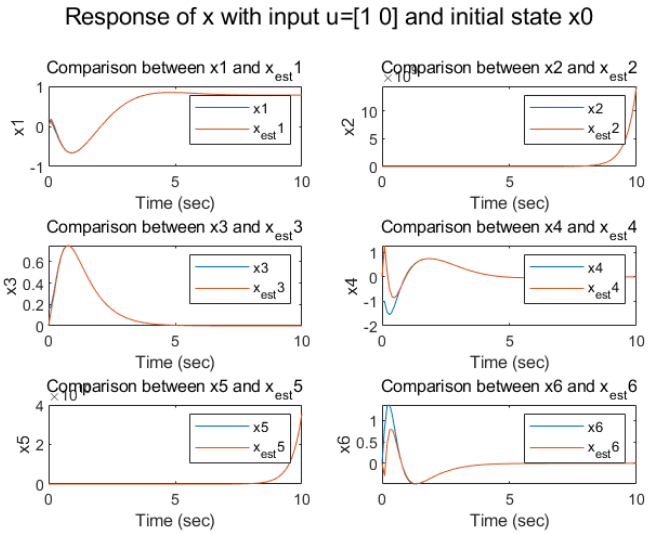

Figure 23: Six state response of the decoupled system (step signal in the first channel)

When designing state feedback controller, we only get two relative degree σ_i to be both 2, which finally leads to the instability. It can also be explained in a view of transfer function. The equivalent feedback transfer function $K(s)$ here actually have an unstable zero-pole cancellation with the original transfer function $G(s)$ so that the closed-loop system shows BIBO stable but not internally stable.

6 Servo control

6.1 Method description

In this section, the three outputs are required to track the set point y_{sp} without steady state error regardless of whether step disturbances are added to input channel. y_{sp} is a constant vector defined by

$$y_{sp} = -\frac{1}{10} CA^{-1}B \begin{pmatrix} -0.5 + (a-b)/20 \\ 0.1 + (b-c)/(a+d+10) \end{pmatrix} \tag{42}$$

First we write the system as

$$\begin{aligned} \dot{x} &= Ax + Bu + B_w w \\ y &= Cx \end{aligned} \tag{43}$$

Since both the setpoint signal and disturbance are step type, an integrator controller is induced to each channel:

$$\int_0^t e(\tau)d\tau = \int_0^t (r - y(\tau))d\tau \tag{44}$$

Then we combine Eq. 43 with Eq. 44 and get an augmented closed-loop system as below

$$\begin{aligned}\begin{pmatrix}\dot{x}\\ \dot{v}\end{pmatrix} &= \begin{pmatrix}A & 0\\ -C & 0\end{pmatrix}\begin{pmatrix}x\\ v\end{pmatrix} + \begin{pmatrix}B\\ 0\end{pmatrix}u + \begin{pmatrix}B_w\\ 0\end{pmatrix}w + \begin{pmatrix}0\\ I\end{pmatrix}r\\ y &= \begin{pmatrix}C & 0\end{pmatrix}\begin{pmatrix}x\\ v\end{pmatrix}\end{aligned} \tag{45}$$

If the original plant is controllable and it satisfies $rank\begin{pmatrix}A & B\\ C & 0\end{pmatrix} = n + m$, where n and m are dimension of state variable and input vector respectively, the augmented system is controllable. Then a PI state feedback control law $u(t) = -K_1x - K_2\int_0^t e(\tau)d\tau$ can be used to stabilize the system. It can also be written in a vector form:

$$u = -K\bar{x} = -\begin{pmatrix}K_1 & K_2\end{pmatrix}\begin{pmatrix}x\\ v\end{pmatrix} \tag{46}$$

Then the resultant closed-loop system becomes

$$\begin{aligned}\begin{pmatrix}\dot{x}\\ \dot{v}\end{pmatrix} &= \begin{pmatrix}A - BK_1 & -BK_2\\ -C & 0\end{pmatrix}\begin{pmatrix}x\\ v\end{pmatrix} + \begin{pmatrix}B_w\\ 0\end{pmatrix}w + \begin{pmatrix}0\\ I\end{pmatrix}r\\ y &= \begin{pmatrix}C & 0\end{pmatrix}\begin{pmatrix}x\\ v\end{pmatrix}\end{aligned} \tag{47}$$

Note that only three cheap sensors can be used to measure the output, we cannot get all six state variables directly. Instead, we use the full-order state observer designed in Section 4 to get estimated $\hat{x}$, thus the control law becomes $u(t) = -K_1\hat{x} - K_2\int_0^t e(\tau)d\tau$. The feedback K can be obtained by former LQR method.

6.2 Simulation results and discussion

The operating set point here is $\begin{pmatrix}0.7398 & 0.7502 & 0.3988\end{pmatrix}^T$ and a step disturbance for the two inputs $w = \begin{pmatrix}-1 & 1\end{pmatrix}^T$ takes effect from $t_d = 10s$. The system model is estabilished in Simulink (Fig. 24), where an observer is used to estimate state variables. First check the set point tracking performance without disturbance. The response signals and tracking errors are illustrated in Fig. 25, where the output become stable and tracking error converges to zero in 6s with small overshoot. The control cost is also monitored and plot in Fig. 26. We can see the first peak is large but the two control cost both come to zero after the system is stabilized.

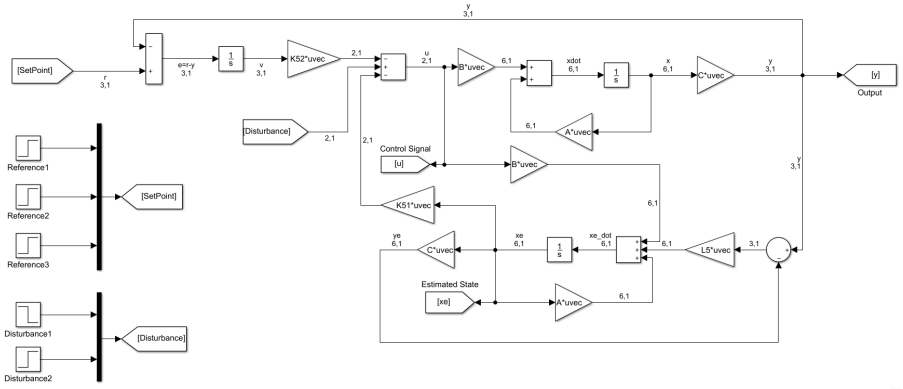

Figure 24: Servo control system

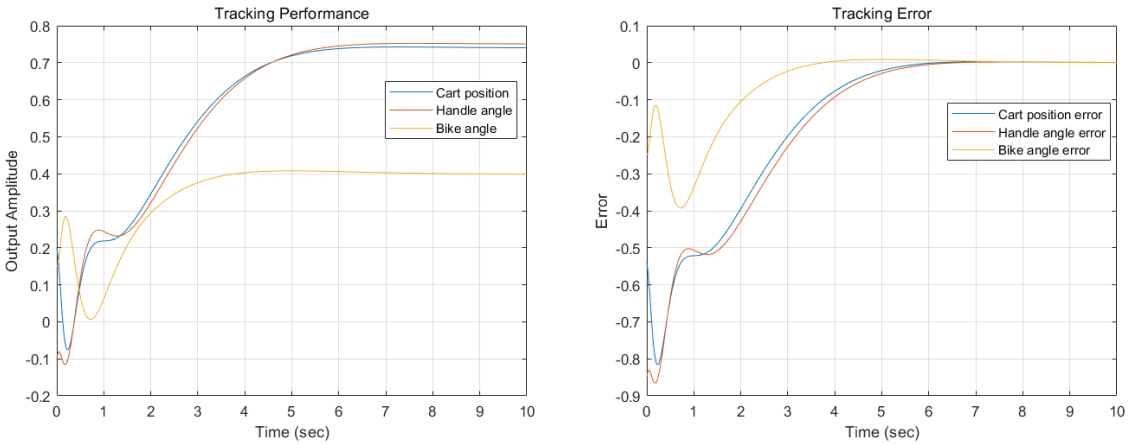

Figure 25: System tracking performance

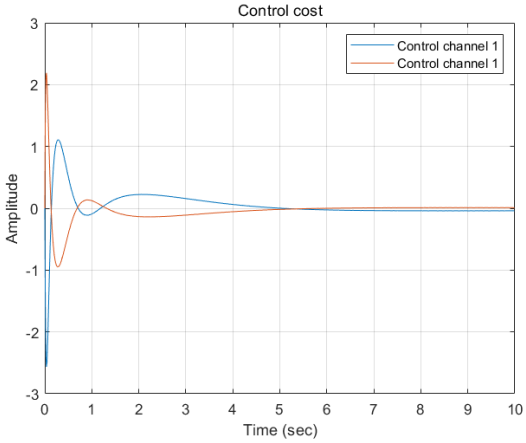

Figure 26: Control cost

Then we add the step disturbance $w = \begin{pmatrix} -1 & 1 \end{pmatrix}^T$ at the two input channels at 10s. The output signal along with tracking error and control signal are given in Fig. 27 and Fig. 28 respectively. The simulation results show that both tracking error and control signal recover to zero after the system is stabilized by the servo controller.

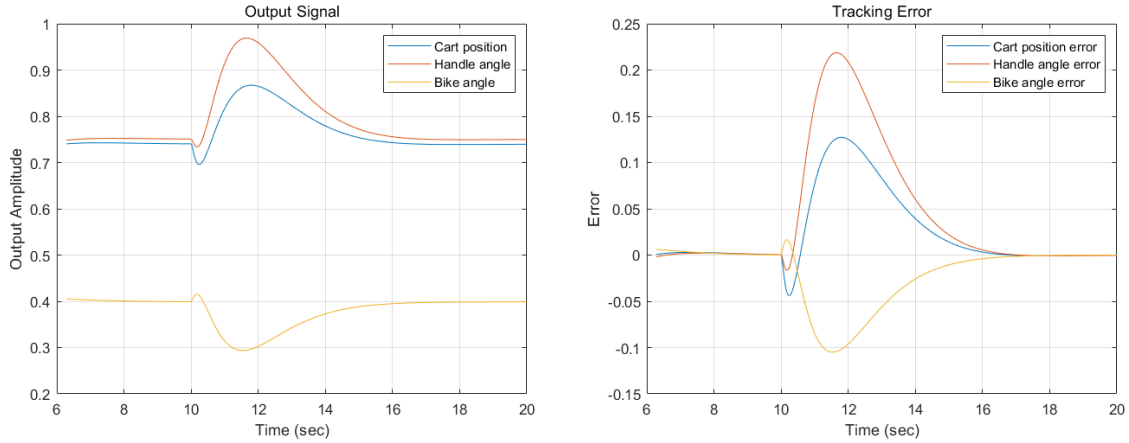

Figure 27: System tracking performance after disturbance

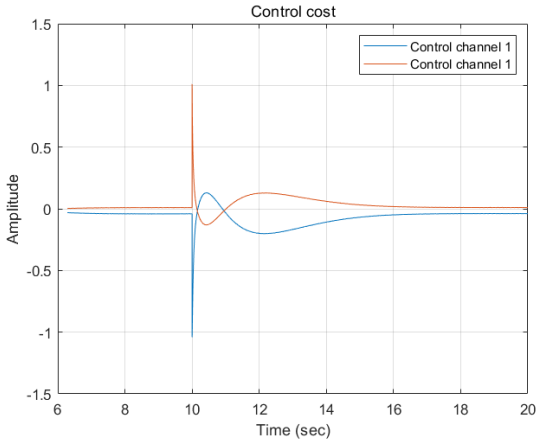

Figure 28: Control cost after disturbance

7 Arbitrary constant set point tracking

7.1 Method description

In this section, we try to change the operating set point y_{sp} and design controllers to track the signal. The newly chosen reference signals are $y_{sp1} = \begin{pmatrix} 3 & 1 & 5 \end{pmatrix}^T$, $y_{sp2} = \begin{pmatrix} 3 & 2 & 5 \end{pmatrix}^T$ and $y_{sp3} = \begin{pmatrix} 30 & 20 & 50 \end{pmatrix}^T$. We use the former multivariable integral control method in last section.

7.2 Simulation results

First for the multivariable integral control method, the tracking peformance for the three different reference signals is shown in Fig. 29. Though the system can be stabilized regardless of step disturbance, zero steady-state error cannot be ensured especially for the last two cases.

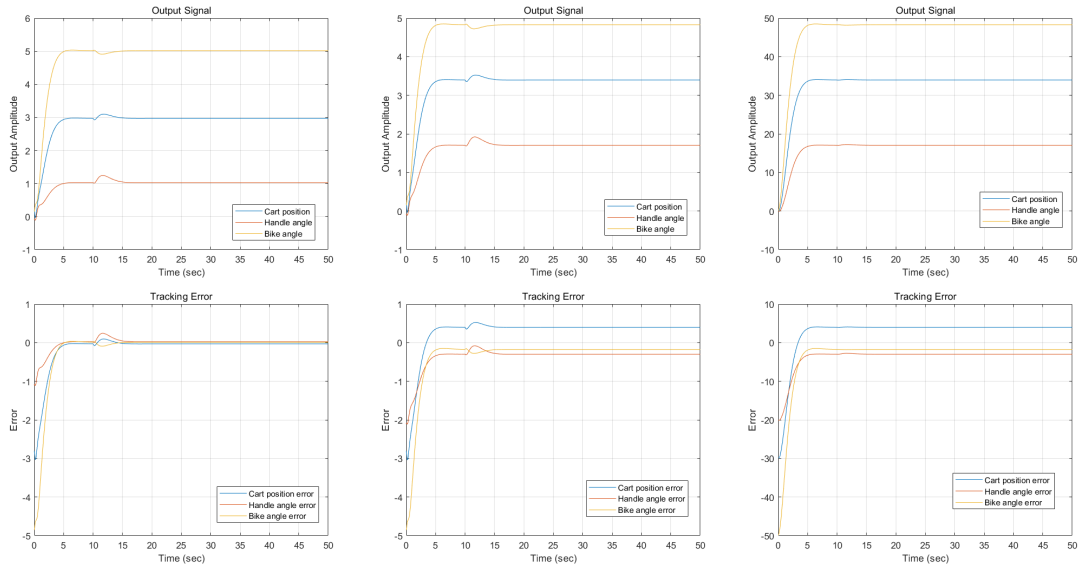

Figure 29: Arbitrary constant tracking performance using multivariable integral controller

7.3 Discussion and explaination

It may be strange that though we use integral controller to track reference signal, the steady-state error increases when the set point is slightly changed. To explain the reaseon, we need to look back on

Eq. 45, where the augmented closed-loop system is controllable only if $rank \begin{pmatrix} A & B \\ C & 0 \end{pmatrix} = n + m$, where n is the dimension of state variables while m is that of both input and output channels. However, our plant here is 2×3 with 2 inputs and 3 outputs. Also check the rank of the controllability matrix, we can find the result is 8 and it is not full row rank. As a result, when using the eigenvalue-based LQR method, we check the Γ matrix in Eq. 21 and find its rank is 17 but not 18, which leads to only 16 stable eigenvectors, i.e.

$$rank(U_{9\times 8}) = rank(V_{9\times 8}) = 8 \tag{48}$$

where $U = \begin{pmatrix} \mu_1 & \cdots & \mu_8 \end{pmatrix}$ and $V = \begin{pmatrix} v_1 & \cdots & v_8 \end{pmatrix}$. Therefore, we cannot let $P = UV^{-1}$ as in Eq. 22 since V is singular and not invertible. It can be easily overlooked because the code written here as "$P = U/V$" in matlab actually do the Overconstrained (Left) Pseudo-Inverse automatically, i.e.

$$P = U(V^T V)^{-1} V^T \tag{49}$$

Left pseudo-inverse can be considered as "best approximation" by minimizing the least squares error, but not the exact solution. Similarly, if use the simple discrete LQR method in Section 3, we can find the matrix P can not converge. Therefore, the multivariable integral controller can not achieve a zero steady-state error. Instead it can only minimize the steady-state error in total and make the system stable regardless of step disturbance.

8 Conclusion

This report test different controllers on the self-balancing vehicles and evalute the simulation results. In the last two sections, we combine LQR state feedback with a state observer to track the reference signal and achieve system stability even with a step disturbance. However, zero steady-state error cannot be ensured for arbitrary constant set point by the multivariable integral contorller, and it needs further refinements in the future.

References

[1] J. Kautsky, N. K. Nichols, and P. Van Dooren, "Robust pole assignment in linear state feedback," *International Journal of control*, vol. 41, no. 5, pp. 1129–1155, 1985.

A Appendix

main_final.m

```
clc,clear;
%   load parameter
[A, B, C, x0] = load_parameter(3, 1, 1, 5);

%   design specifications check

D = zeros(3, 2);
t = 0 : 0.1 : 10;

u1 = [ones(size(t,2),1),zeros(size(t,2),1)];
u2 = [zeros(size(t,2),1),ones(size(t,2),1)];
u0 = [zeros(size(t,2),1),zeros(size(t,2),1)];
u3 = [ones(size(t,2),1),ones(size(t,2),1)];

%% Q1
%   pole placement feedback
%       Assume that you can measure all the six state variables, design a state feedback controller using
%   the pole place method, simulate the designed system and show all the six state responses to non-
%   zero initial state with zero external inputs. Discuss effects of the positions of the poles on system
%   performance, and also monitor control signal size. In this step, both the disturbance and set point
%   can be assumed to be zero.

damp = 0.8;
wn = 3.125;
lamda1 = -(damp*wn + wn*sqrt(1-damp*damp)*1i);
lamda2 = -(damp*wn - wn*sqrt(1-damp*damp)*1i);
lamda3 = -4*damp*wn;
lamda4 = -4.3*damp*wn;
lamda5 = -4.6*damp*wn;
lamda6 = -5.0*damp*wn;
p = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6];

[K11,K1] = pole_placement(A,B,p);

% check for K11
ss_q1 = ss(A-B*K11, B, C, D);
global_check(ss_q1,u1,u2,t,zeros(6,1));

% check for K1
ss_q1 = ss(A-B*K1, B, C, D);
global_check(ss_q1,u1,u2,t,zeros(6,1));

% task 1 check
task12_check(ss_q1,t,u0,x0,K11);

%% Q2
%   LQR control
%       Assume that you can measure all the six state variables, design a state feedback controller using
%   the LQR method, simulate the designed system and show all the state responses to non-zero
%   initial state with zero external inputs. Discuss effects of weightings Q and R on system
%   performance, and also monitor control signal size. In this step, both the disturbance
```

```
     and set point
%    can be assumed to be zero.

Q = [ 0.2   0   0   0   0   0;
      0   1   0   0   0   0;
      0   0   1   0   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]*1;

%   eigenvalue eigenvector method in class
K2 = lqr_control(A,B,Q,R);

%   discrete lqr
K22=dlqr_control(A,B,Q,R);

% check for K22
ss_q2 = ss(A-B*K22, B, C, D);
global_check(ss_q2,u1,u2,t,zeros(6,1));

% check for K2
ss_q2 = ss(A-B*K2, B, C, D);
global_check(ss_q2,u1,u2,t,zeros(6,1));

% task 2 check
task12_check(ss_q2,t,u0,x0,K2);

%% Q3
% Assume you can only measure the three outputs.
% Design a state observer, simulate the resultant observer-based LQR control system,
% monitor the state estimation error, investigate effects of observer poles
% on state estimation error and closed-loop control performance. In this step, both
% the disturbance and set point can be assumed to be zero. (10 points)
Q = [ 0.2   0   0   0   0   0;
      0   1.1   0   0   0   0;
      0   0   1.1   0   0   0;
      0   0   0   0.2   0   0;
      0   0   0   0   1.1   0;
      0   0   0   0   0   1.1]*20;
R = [1   0;
     0   1]*5;

%   the lqr feedback
K3=lqr_control(A,B,Q,R);

damp = 0.8;
wn=2;
lamda1 = -(damp*wn + wn*sqrt(1-damp*damp)*1i);
lamda2 = -(damp*wn - wn*sqrt(1-damp*damp)*1i);
lamda3 = -4*damp*wn;
lamda4 = -4.3*damp*wn;
lamda5 = -4.6*damp*wn;
lamda6 = -5.0*damp*wn;
p1 = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6]*2.7;
p2 = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6]*2;
p3 = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6]*4;

[L,At,Bt,Ct]=state_observer(K3,A,B,C,p1);
ss_q3=ss(At,Bt,Ct,D);
```

```
112
113 [L,At,Bt,Ct]=state_observer(K3,A,B,C,p2);
114 ss_q32=ss(At,Bt,Ct,D);
115
116 [L,At,Bt,Ct]=state_observer(K3,A,B,C,p3);
117 ss_q33=ss(At,Bt,Ct,D);
118
119 xe=x0;
120
121 % global check
122 global_check(ss_q3,u1,u2,t,[zeros(6,1);xe]);
123
124 % task 3 check
125 task3_check(ss_q3,ss_q32,ss_q33,u0,t,[x0;xe]);
126
127 %% Q4
128 %  Design a decoupling controller with closed-loop stability and simulate the
129 %  step set point response of the resultant control system to verify
130 %  decoupling performance with stability. In this step, the disturbance can be
131 %  assumed to be zero. Is the decoupled system internally stable?
132 %  Please provide both the step (transient) response with zero initial states
133 %  and the initial response with respect to x0 of the decoupled system
134 %  to support your conclusion.
135 C2=[1 0 0 0 0 0; 0 0 1 0 0 0];
136 D2=[0 0;0 0];
137
138 %   state feedback
139 [K4,F4]=decoupler_sf(A,B,C2);
140
141 ss_q4 = ss(A-B*K4, B*F4, C2, D2);
142
143 % global check
144 global_check(ss_q4,u1,u2,t,zeros(6,1));
145
146 % task 4 check
147 task4_check(ss_q4,t,u0,u1,u2,u3,x0);
148
149 %% Q5
150 % Assume that you only have three cheap sensors to measure the output. Design a
      controller such
151 % that the plant (vehicle) can operate around the set point as close as possible at
      steady state even
152 % when step disturbances are present at the plant input. Plot out both the control and
      output signals.
153 % In your simulation, you may assume the step disturbance for the two inputs,
154 % takes effect from time td = 10s afterwards.
155
156 ysp=-0.1*C/A*B*[-0.5+(3-1)/20; 0.1+(1-1)/(3+5+10)];
157 % ysp=[1;3;5];
158 [K5,L5]=servo(A,B,C);
159 K51=K5(:,1:6);
160 K52=K5(:,7:9);
161
162 %%
163 % run simulink
164 % plot
165 figure;
166 curve=plot(yout.time(300:end,1),yout.signals(1).values(300:end,1),yout.time(300:end,1),
      yout.signals(1).values(300:end,2),yout.time(300:end,1),yout.signals(1).values(300:end
      ,3));
167 grid on;
```

```
168 legend('Cart position','Handle angle','Bike angle');
169 xlabel('Time (sec)'), ylabel('Output Amplitude');
170 title('Output Signal');
171
172 figure;
173 curve=plot(yout.time(300:end,1),yout.signals(1).values(300:end,1)-ysp(1),yout.time(300:
        end,1),yout.signals(1).values(300:end,2)-ysp(2),yout.time(300:end,1),yout.signals(1).
        values(300:end,3)-ysp(3));
174 grid on;
175 legend('Cart position error','Handle angle error','Bike angle error');
176 xlabel('Time (sec)'), ylabel('Error');
177 title('Tracking Error');
178
179 figure;
180 curve=plot(controlSignal.time(300:end,1),controlSignal.signals(1).values(300:end,1),
        controlSignal.time(300:end,1),controlSignal.signals(1).values(300:end,2));
181 grid on;
182 legend('Control channel 1','Control channel 1');
183 xlabel('Time (sec)'), ylabel('Amplitude');
184 title('Control cost');
```

main_final.m

load_parameter.m

```
1  function [A, B, C, x0] = load_parameter(a, b, c, d)
2  %   parameter a, b, c, d
3  % a = 3;
4  % b = 1;
5  % c = 1;
6  % d = 5;
7
8  %   alpha, beta, gamma, delta
9  alpha = 15.5 - a / 3 + b / 2;
10 beta = 27.5 - d / 2;
11 gamma = 11.5 + (a - c) / (b + d + 3);
12 delta = 60 + (a - b) * c / 10;
13
14 %   g
15 g = 9.8;
16
17 %   Mass of each part
18 M_f = 2.14 + c / 20;
19 M_r = 5.91 - b / 10;
20 M_c = 1.74;
21
22 %   Vertical length from a floor to a center-of-gravity of each part
23 H_f = 0.18;
24 H_r = 0.161;
25 H_c = 0.098;
26
27 %   Horizontal length from a front wheel rotation axis to a center-of-gravity of
28 %   part of front wheel and steering axis.
29 L_Ff = 0.05;
30 L_F = 0.133;
31
32 %   Horizontal length from a rear wheel rotation axis to a center-of-gravity of
33 %   part of rear wheel and steering axis.
34 L_r = 0.128;
35 L_R = 0.308 + (a - d) / 100;
36
37 %   Horizontal length from a rear wheel rotation axis to a center-of-gravity of the cart
```

```
    system
L_c = 0.259;

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

%   Moment of inertia for part of front wheel z axially.
% J_fz =

%   Moment of inertia for part of rear wheel that contains cart system z axially.
% J_z

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

%   Viscous coefficient for part of front wheel around z axis.
% miu_fz

%   Viscous coefficient for part of rear wheel that contains cart system around z axis.
% miu_z

%   A viscosity coefficient of a movement direction of the cart system
% miu_c

%   Part of front wheel, rear wheel, and cart system respectively
% f
% r
% c

%   Cart position, handle angle and bike angle
% d_t
% phi_t
% psai_t

%   parameter in matrix
den = M_f * H_f * H_f + M_r * H_r * H_r + M_c * H_c * H_c + J_x;
a51 = - M_c * g / den;
a52 = (M_f * H_f + M_r * H_r + M_c * H_c) * g / den;
a53 = (M_r * L_r * L_F + M_c * L_c * L_F + M_f * L_Ff * L_R) * g / (L_R + L_F) / den;
a54 = - M_c * H_c * alpha / den;
a55 = - miu_x / den;
a56 = M_f * H_f * L_Ff * gamma / den;
b51 = M_c * H_c * beta / den;
b52 = - 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;
    a51    a52     a53     a54     a55     a56;
     5    -3.6       0       0       0   -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];

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

end
```

load_parameter.m

pole_placement.m

```
function [K11,K] = pole_placement(A, B, p)

syms s;

% lamda1 = -1.25 + 1.875*1i;
% lamda2 = -1.25 - 1.875*1i;
% lamda3 = -5;
% lamda4 = -5.375;
% lamda5 = -5.75;
% lamda6 = -6.25;
%
% lamda1 = -5 + 1.875*1i;
% lamda2 = -5 - 1.875*1i;
% lamda3 = -20;
% lamda4 = -21.5;
% lamda5 = -23;
% lamda6 = -25;

polynomial = (s-p(1))*(s-p(2))*(s-p(3))*(s-p(4))*(s-p(5))*(s-p(6));
pol_cof = double(coeffs(polynomial));

K11 = place(A,B,p);
% K11=acker(A,B,p);

dim=size(B,2);

if dim==2

    K=zeros(2,6);

    Ad = [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;
          -pol_cof(1 : 6)];

    Ad = [0 1 0            0 0 0;
          0 0 1            0 0 0;
          -pol_cof(1:3)    0 0 0;
          0 0 0            0 1 0;
          0 0 0            0 0 1;
          0 0 0            -pol_cof(4:6)];

    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];
    Wc = [B A*B A^2*B A^3*B A^4*B A^5*B];
%     assert(rank(Wc(:, 1:6)) == 6);
    CC = Wc(:, 1:6);
    CC = [CC(:,1), CC(:,3), CC(:,5), CC(:,2), CC(:,4), CC(:,6)];
    C_inv = inv(CC);
```

```
    d1 = 3;
    d2 = 3;
    T = [C_inv(d1,:);
         C_inv(d1,:)*A;
         C_inv(d1,:)*A*A;
         C_inv(d1+d2,:);
         C_inv(d1+d2,:)*A;
         C_inv(d1+d2,:)*A*A];
    Aba = T*A/T;
    Bba = T*B;
    % Aba = round(Aba);
    % Bba = round(Bba);
    Aba(abs(Aba) < 1e-5) = 0;
    Bba(abs(Bba) < 1e-5) = 0;

    K_num = solve(Ad == Aba-Bba*K);
    K_ans = double(struct2array(K_num));
    Kba = [K_ans(1:6);
           K_ans(7:12)];
    K = Kba * T;

elseif dim==3

    Ad = [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;
          -pol_cof(1 : 6)];

    Ad = [0 1 0           0 0 0;
          0 0 1           0 0 0;
          -pol_cof(1:3)   0 0 0;
          0 0 0           0 1 0;
          0 0 0           0 0 1;
          0 0 0           -pol_cof(4:6)];

    syms k11 k12 k13 k14 k15 k16 k21 k22 k23 k24 k25 k26 k31 k32 k33 k34 k35 k36;
    K = [k11 k12 k13 k14 k15 k16;
         k21 k22 k23 k24 k25 k26;
         k31 k32 k33 k34 k35 k36];
    Wc = [B A*B A^2*B A^3*B A^4*B A^5*B];
%     assert(rank(Wc(:, 1:6)) == 6);
    CC = Wc(:, 1:6);
    CC = [CC(:,1), CC(:,3), CC(:,5), CC(:,2), CC(:,4), CC(:,6)];
    C_inv = inv(CC);
    d1 = 3;
    d2 = 3;
    T = [C_inv(d1,:);
         C_inv(d1,:)*A;
         C_inv(d1,:)*A*A;
         C_inv(d1+d2,:);
         C_inv(d1+d2,:)*A;
         C_inv(d1+d2,:)*A*A];
    if rank(T)<6
        K=place(A,B,p);
        return;
    end
    Aba = T*A/T;
    Bba = T*B;
    Aba(abs(Aba) < 1e-5) = 0;
```

```
    Bba(abs(Bba) < 1e-5) = 0;

    K_num = solve(Ad == Aba-Bba*K);
    K_ans = double(struct2array(K_num));
    Kba = [K_ans(1:6);
           K_ans(7:12);
           K_ans(13:18)];
    K = Kba * T;

end

end
```

pole_placement.m

lqr_control.m

```
function [K] = lqr_control(A,B,Q,R)
%LQR_CONTROL
% Q = [ 0.2   0   0   0   0   0;
%       0   1   0   0   0   0;
%       0   0   1   0   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]*1;
M = [ A   -B/R*B';
     -Q   -A'];

[evec, eval] = eig(M);
eval = sum(eval);
evec_stable = evec(:,find(real(eval)<0));
% P = evec_stable(,:)
V = evec_stable(1:6,:);
U = evec_stable(7:12,:);
P = U/V;

K = R \ B' * P;

end
```

lqr_control.m

dlqr_control.m

```
function [K] = dlqr_control(A,B,Q,R)
%DLQR_CONTROL Summary of this function goes here
%   Detailed explanation goes here
%   discretize state space equation
m=size(A,1);
n=size(B,2);
dt=0.01;
Aba=(eye(m)-A*dt/2)\(eye(m)+A*dt/2);
Bba=B*dt;
P=Q;
P_new=Q+Aba'*P*Aba-Aba'*P*Bba/(R+Bba'*P*Bba)*Bba'*P*Aba;
it_count=1;
while max(abs(P_new-P),[],'all') > 1e-5
    P=P_new;
    P_new=Q+Aba'*P*Aba-Aba'*P*Bba/(R+Bba'*P*Bba)*Bba'*P*Aba;
    it_count=it_count+1;
end
```

```
K=inv(R+Bba'*P_new*Bba)*Bba'*P_new*Aba;
end
```

dlqr_control.m

state_observer.m

```
function [L,At,Bt,Ct] = state_observer(K,A,B,C,p)
%STATE_OBSERVER Summary of this function goes here
%   Detailed explanation goes here

[L1,L]=pole_placement(A',C',p);
L=L';

At=[A-B*K           B*K;
    zeros(size(A))  A-L*C];
Bt=[B;zeros(size(B))];
Ct=[C   zeros(size(C))];
end
```

state_observer.m

decoupler_sf.m

```
function [K,F] = decoupler_sf(A,B,C)
%DECOUPLER
m=size(A,1);
r=size(B,2);
n=size(C,1);
% syms s;
% G = C/(s*eye(m)-A)*B;

sigma=zeros(n,1);
for i=1:n
    count=1;
    for j=1:m
        if C(i,:)*(A^(count-1))*B ~= 0
            break;
        else
            count=count+1;
        end
    end
    if count == m+1
        sigma(i)=m;
    else
        sigma(i)=count;
    end
end

Bstar=zeros(n,r);
Cstar=zeros(n,m);
for i=1:n
    Bstar(i,:)=C(i,:)*(A^(sigma(i)-1))*B;
    Cstar(i,:)=C(i,:)* A^(sigma(i));
end
F=inv(Bstar);
% K=Bstar\Cstar;

damp = 0.707;
wn = 1.13;
lamda1 = -damp*wn + wn*sqrt(1-damp*damp)*1i;
lamda2 = -damp*wn - wn*sqrt(1-damp*damp)*1i;
lamda3 = -1.6;
```

```
lamda4 = -1.7;

fA1 = (A-lamda1*eye(m))*(A-lamda2*eye(m));
fA2 = (A-lamda3*eye(m))*(A-lamda4*eye(m));

Csstar=zeros(n,m);
% for i=1:n
% %     Csstar(i,:)=C(i,:)*fA;
%     Csstar(i,:)=C(i,:)*polinomial(i);
% end
Csstar(1,:)=C(1,:)*fA1;
Csstar(2,:)=C(2,:)*fA2;
K=F*Csstar;


end
```

decoupler_sf.m

servo.m

```
function [K,L] = servo(A,B,C)
%SERVO
Aba=[ A  zeros(6,3);
     -C  zeros(3,3)];
Bba=[B; zeros(3,2)];
Bwba=[B; zeros(3,2)];
Brba=[zeros(6,3); eye(3)];
Cba=[C zeros(3,3)];

Qc=[Bba Aba*Bba Aba^2*Bba Aba^3*Bba Aba^4*Bba Aba^5*Bba Aba^6*Bba Aba^7*Bba Aba^8*Bba];
rank(Qc)
rank([A B;C zeros(3,2)])

%dlqr
Q=eye(9)*20;
% Q=[2  0  0  0  0  0  0  0  0;
%    0  2  0  0  0  0  0  0  0;
%    0  0  3  0  0  0  0  0  0;
%    0  0  0  4  0  0  0  0  0;
%    0  0  0  0  5  0  0  0  0;
%    0  0  0  0  0  4  0  0  0;
%    0  0  0  0  0  0  3  0  0;
%    0  0  0  0  0  0  0  2  0;
%    0  0  0  0  0  0  0  0  2]*20;
R=[1 0;
   0 1]*5;
K=lqr(Aba,Bba,Q,R,0);
% K=lqr_control(Aba,Bba,Q,R);

M = [ Aba   -Bba/R*Bba';
     -Q   -Aba'];

[evec, eval] = eig(M);
eval = sum(eval);
evec_stable = evec(:,find(real(eval)<0));
% P = evec_stable(,:)
V = evec_stable(1:9,:);
U = evec_stable(10:18,:);
P = U/V;

K = R \ Bba' * P;
```

```
42
43
44 K1=K(:,1:6);
45 K2=K(:,7:9);
46
47 %observer
48 % lamda1 = -10+10*1i;
49 % lamda2 = -10-10*1i;
50 % lamda3 = -50;
51 % lamda4 = -75;
52 % lamda5 = -100;
53 % lamda6 = -125;
54 % p = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6]*2.7;
55
56 damp = 0.8;
57 wn=2;
58 lamda1 = -(damp*wn + wn*sqrt(1-damp*damp)*1i);
59 lamda2 = -(damp*wn - wn*sqrt(1-damp*damp)*1i);
60 lamda3 = -4*damp*wn;
61 lamda4 = -4.3*damp*wn;
62 lamda5 = -4.6*damp*wn;
63 lamda6 = -5.0*damp*wn;
64 p = [lamda1 lamda2 lamda3 lamda4 lamda5 lamda6]*2.7;
65 [L,At,Bt,Ct]=state_observer(K1,A,B,C,p);
66
67 end
```

servo.m

global_check.m

```
1  function [] = global_check(ss,u1,u2,t,x0)
2  %UNTITLED Summary of this function goes here
3  %   Detailed explanation goes here
4
5  %   u=[1 0]
6  [y,t,x]=lsim(ss,u1,t,x0);
7  q1_info = stepinfo(y,t);
8
9  if size(y,2)==2
10     figure;
11     for i = 1:2
12         plot(t,y(:,i));
13         if i==1
14             legend_str{i} = ['y1: overshoot: ', num2str(q1_info(i).Overshoot),'%, ', ...
15                         '2% setting time: ', num2str(q1_info(i).SettlingTime),'s'];
16         else
17             legend_str{i} = 'y2';
18         end
19         hold on;
20     end
21     legend(legend_str);
22     xlabel('Time (sec)');
23     ylabel('y');
24     sgtitle('Output response with input u=[1 0]');
25
26 else
27
28     figure;
29     subplot(3,1,1);
30     plot(t,y(:,1));
31     title({'Output curve with u=[1 0]', ...
```

```
        ['overshoot: ', num2str(q1_info(1).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(1).SettlingTime),'s' ]});
    xlabel('Time (sec)');
    ylabel('Cart Position (m)');

    subplot(3,1,2);
    plot(t,y(:,2));
    title({'Output curve with u=[1 0]', ...
        ['overshoot: ', num2str(q1_info(2).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(2).SettlingTime),'s' ]});
    xlabel('Time (sec)');
    ylabel('Handle Angle (rad)');

    subplot(3,1,3);
    plot(t,y(:,3));
    title({'Output curve with u=[1 0]', ...
        ['overshoot: ', num2str(q1_info(3).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(3).SettlingTime),'s' ]});
    xlabel('Time (sec)');
    ylabel('Bike Angle (rad)');
end


%   u=[0 1]
[y,t,x]=lsim(ss,u2,t,x0);
q1_info = stepinfo(y,t);

if size(y,2)==2

    figure;
    for i = 1:2
        plot(t,y(:,i));
        if i==2
            legend_str{i} = ['y2: overshoot: ', num2str(q1_info(i).Overshoot),'%, ', ...
                        '2% setting time: ', num2str(q1_info(i).SettlingTime),'s'];
        else
            legend_str{i} = 'y1';
        end
        hold on;
    end
    legend(legend_str);
    xlabel('Time (sec)');
    ylabel('y');
    sgtitle('Output response with input u=[0 1]');

else

    figure;
    subplot(3,1,1);
    plot(t,y(:,1));
    title({'Output curve with u=[0 1]', ...
        ['overshoot: ', num2str(q1_info(1).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(1).SettlingTime),'s' ]});
    xlabel('Time (sec)');
    ylabel('Cart Position (m)');

    subplot(3,1,2);
    plot(t,y(:,2));
    title({'Output curve with u=[0 1]', ...
        ['overshoot: ', num2str(q1_info(2).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(2).SettlingTime),'s' ]});
```

```
    xlabel('Time (sec)');
    ylabel('Handle Angle (rad)');

    subplot(3,1,3);
    plot(t,y(:,3));
    title({'Output curve with u=[0 1]', ...
        ['overshoot: ', num2str(q1_info(3).Overshoot),'%'], ...
        ['2% setting time: ', num2str(q1_info(3).SettlingTime),'s' ]});
    xlabel('Time (sec)');
    ylabel('Bike Angle (rad)');
end

end
```

global_check.m

task12_check.m

```
function [] = task12_check(ss,t,u0,x0,K11)
%TASK1_CHECK Summary of this function goes here
%   Detailed explanation goes here

% 6 state responses to x0 with zero inputs
[y,t,x]=lsim(ss,u0,t,x0);

figure;
for i = 1:6
    subplot(3,2,i);
    plot(t,x(:,i));
    title(['Response for x',num2str(i)]);
    xlabel('Time (sec)');
    ylabel(['x',num2str(i)]);
end
sgtitle('Six state responses with zero inputs');

% monitor the control signal size
u=-x*K11';
figure;
for i = 1:2
    subplot(2,1,i);
    plot(t,u(:,i));
    title(['Control signal u',num2str(i)]);
    xlabel('Time (sec)');
    ylabel(['u',num2str(i)]);
end
% legend(legend_str);
sgtitle('Control signal size');

end
```

task12_check.m

task3_check.m

```
function [] = task3_check(ss1,ss2,ss3,u0,t,x0)
%TASK3_CHECK Summary of this function goes here
%   Detailed explanation goes here

% pole influence
[y,t,x]=lsim(ss1,u0,t,x0);
[ydot1,t,xdot1]=lsim(ss2,u0,t,x0);
[ydot2,t,xdot2]=lsim(ss3,u0,t,x0);
figure;
```

```
for i = 1:3
    subplot(3,1,i);
    plot(t,y(:,i),'-r',t,ydot1(:,i),'-g',t,ydot2(:,i),'b');
    legend('3 times faster','2 times faster','4 times faster');
    title(['Response for y',num2str(i)]);
    xlabel('Time (sec)');
    ylabel(['y',num2str(i)]);
end
sgtitle('System performance with different observer poles');

% monitor the state estimation error
t1=0:0.05:1.2;
u00 = [zeros(size(t1,2),1),zeros(size(t1,2),1)];

[y,t1,x]=lsim(ss1,u00,t1,x0);

e=x(:,7:end);
x=x(:,1:6);
x_est=x-e;

figure;
for i = 1:6
    subplot(3,2,i);
    plot(t1,x(:,i));
    hold on;
    plot(t1,x_est(:,i));
    legend(['x',num2str(i)],['x_{est}',num2str(i)]);
    title(['Comparison between x',num2str(i),' and x_{est}',num2str(i)]);
    xlabel('Time (sec)');
    ylabel(['x',num2str(i)]);
end
sgtitle('Comparison between x and x_{est}');

figure;
e1=e(:,1);e2=e(:,2);e3=e(:,3);e4=e(:,4);e5=e(:,5);e6=e(:,6);
plot(t1,e1,'-r', t1,e2,'-b', t1,e3,'-g', t1,e4,'-y', t1,e5,'-c', t1,e6,'-k');
legend('e1','e2','e3','e4','e5','e6');
xlabel('Time (sec)');
ylabel('State estimation error');
title('State estimation error with time');


end
```

task3_check.m

task4_check.m

```
function [] = task4_check(ss,t,u0,u1,u2,u3,x0)
%TASK4_CHECK Summary of this function goes here
%   Detailed explanation goes here

%   check if external stable (x0)

% u1=[1 0]
[y, t, x] = lsim(ss, u1, t, x0);

figure;
for i = 1:2
    plot(t,y(:,i));
    legend_str{i} = ['y',num2str(i)];
    hold on;
```

```
end
legend(legend_str);
xlabel('Time (sec)');
ylabel('y');
sgtitle('Output response with input u=[1 0]');

% u2=[0 1]
[y, t, x] = lsim(ss, u2, t, x0);

figure;
for i = 1:2
    plot(t,y(:,i));
    legend_str{i} = ['y',num2str(i)];
    hold on;
end
legend(legend_str);
xlabel('Time (sec)');
ylabel('y');
sgtitle('Output response with input u=[0 1]');

% u3=[1 1]
[y, t, x] = lsim(ss, u3, t, x0);

figure;
for i = 1:2
    plot(t,y(:,i));
    legend_str{i} = ['y',num2str(i)];
    hold on;
end
legend(legend_str);
xlabel('Time (sec)');
ylabel('y');
sgtitle('Output response with input u=[1 1]');


% u3=[0 0]
[y, t, x] = lsim(ss, u0, t, x0);

figure;
for i = 1:2
    plot(t,y(:,i));
    legend_str{i} = ['y',num2str(i)];
    hold on;
end
legend(legend_str);
xlabel('Time (sec)');
ylabel('y');
sgtitle('Output response with input u=[0 0]');

%   check if internally stable
p=pole(ss);
for i=1:size(p)
    if real(p(i))>0
        disp(['Find pole with negative part:',num2str(p(i))]);
    end
end

end
```

task4_check.m