Self-Balanced Cube Robot Modeling and Control

Qixiao Zhang
Electrical Engineering, Columbia University
New York, USA
qz2487@columbia.edu

***Abstract*—This paper is for the course project of EE6602: Modern Control Theory. In this paper, the model of this self-balanced cube robot is proposed and we used several ways to control the reaction wheels inside the robot so that the robot can stand on its corner. Some simplification to the model is deployed in the modelling to reduce the complexity of computation.**

***Index Terms*—Contorl, H_∞, LQR**

I. Introduction

With the development of control methods, we can control the robots more and more accurate so that they can meet our needs in daily lives and in industry. A fine-controlled robot can do very dexterously jobs. Therefore, I want to do this applied projects about robot controlling.

As we know, a very classic model in mechanical system about self-balancing is the famous "inverted pendulum". This is a start of self-balanced system in the control field. Self-balanced system has a very wide range of applications in our daily life and the most famous one is the self-balanced car. With the similar idea, a cube robot called "Cubli" [1] which can stand on its corner attracts me. This is a self-balanced cubic robot which can balance on one of its corner. Besides, it can also 'work' or 'jump up'. To make the cube stand on its corner, there are a few inner reaction wheels which can balance the cube itself. Therefore, we have to design an appropriate controller to generate a proper torque inside the cube to maintain its balance via the wheel. And this is the start of my idea on this project.

At the very beginning, my idea was to simplify this robot so that it can stand on its edge. But after deriving the math model of this robot, I changed my idea since this model is way too easy to control. It just need one reaction wheel and therefore there's only one input and two states. Then I switched my idea to imitate the robot standing on the corner and the difficulty to model this robot increase dramatically.

The specific problems will be demonstrated in Modeling Section (III).

II. Literature Review

I found a few papers about the self-balancing cube robot. I used the model they provided but I changed a lot during the induction of the modeling because there are a lot of typos and inconsistency in these paper. For example, in paper [2], the model they gave shows that they mixed up the $\ddot{\gamma}$ and γ. They have the output quantity $y = [\alpha \quad \beta \quad \gamma \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3]^T$. However, because we only know the representation of $\ddot{\gamma}$ and it is not depend on $\dot{\gamma}$ or γ, so that we actually cannot observe the status of γ since we can only know the acceleration of γ. An astonishing part is that in the end they managed to control the angle γ, which is not consistent to their preprocessing.

Another paper [3] about this same model has the same problem. After I derived this model, I cannot found a way to control the yaw angle γ of this robot with three actuators. However, after fixing the problems in their paper, this model is very reasonable and acceptable since it can make sure the cube stand on the corner even though we cannot control the yaw angle.

III. Modeling with Lagrangian Equation of Motion

We used the model from paper [2] and [3]. This is a cube with three reactions wheels embedded inside. In our design, we are going to adjust the voltage of three motors to control the torque of the whole system.

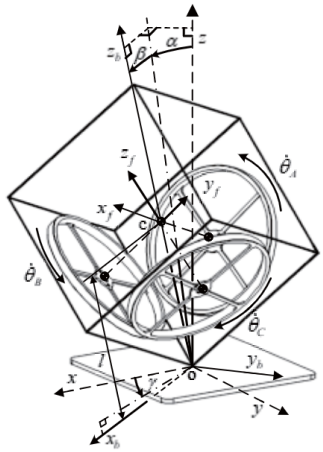

Fig. 1. Schematic diagram of Cube robot

The model is shown in Fig 1. And we are going to analyze this system with Lagrangian Equation of Motion.

First, we can assume these parameters in the system:

Constant:

l: centroid height of the system.

m: mass of the robot.

I_x,I_y,I_z: The total inertia moment of the system in the body frame.

I_w: The inertia moment of the reaction wheels.

K_t: Moment constant of motor.
R_m: Armature resistance.
g: Gravity constant.

State Variable:

α,β,γ: The deflection angles in reference frame in pitch, roll and yaw direction.

$\dot{\theta}_A, \dot{\theta}_B, \dot{\theta}_C$: The angle velocities of reaction wheels in reaction wheels coordinate frame.

Input:

v_a,v_b,v_c: Input voltages of motors respectively.

A. Coordinate system

As mentioned in [3], we have three coordinates: $o-xyz$, $o-x_by_bz_b$ and $c-x_fy_fz_f$. The first one is the fixed coordinate, the second one is the body coordinate, the third one is the reaction wheel coordinate. The relationship between these coordinates is shown below.

The transformation matrix we will use later from body to reaction wheel coordinate is

$$T_{bf} = R_y(-35.3°) \cdot R_x(45°) \tag{1}$$

where R_y and R_x denote the rotation matrix in y and x axis.

B. Energy of the System and Lagrange Function

To construct the Lagrange equation of motion, we need to construct the Lagrange function first. We need the total energy of the whole system. We can compute it with these equations:

- Kinetic energy of rotation for the robot.

$$T_{robot} = \frac{1}{2}(I_y\ddot{\alpha}^2 + I_x\ddot{\beta}^2 + I_z\ddot{\gamma}^2) \tag{2}$$

- Kinetic energy of rotation for the wheels.

$$T_{wheel} = \frac{1}{2}I_w(\ddot{\theta}_A^2 + \ddot{\theta}_B^2 + \ddot{\theta}_C^2) \tag{3}$$

- Potential energy of the system.

$$E = mgl\cos\alpha\cos\beta \tag{4}$$

From equation (2), (3), (4), we can derive our Lagrange Function:

$$L = T_{robot} + T_{wheel} - E \tag{5}$$

C. Armature Circuit Analysis

We are using Brushless DC motors to control the torque of the reaction wheels. we can use the amature circuit equation to derive the voltage and then get the relationship between torque τ and amature voltage. The amature circuit equation:

$$v_j = R_m i_j + L_m\frac{di}{dt} + Ke\dot{\theta}_j, \quad j = A, B, C$$

where v_j is the input voltage of motors, i_j is the current of motors, L_m is the amature capacity, K_e is the back electromotive force constant.

Since L_m is very small, we can ignore that. And the generated torque is:

$$\tau_j = K_t i = K_t(v_j - K_e\dot{\theta}_j)$$

Also, in BLDC, $K_t = K_e$. Therefore, we can get:

$$\tau_j = \frac{K_t v_j}{R_m} - \frac{K_t^2\dot{\theta}_j}{R_m} \tag{6}$$

D. Lagrange Equation of Motion

The Lagrange equation of motion is:

$$\frac{d}{dt}(\frac{\partial L}{\partial\dot{q}_i}) - \frac{\partial L}{\partial q_i} = \tau_i \tag{7}$$

where $q_1 = \alpha$, $q_2 = \beta$, $q_3 = \gamma$, $q_4 = \theta_A$, $q_5 = \theta_B$, $q_6 = \theta_C$,

And $\tau_1 = \tau_\alpha$, $\tau_2 = \tau_\beta$, $\tau_3 = \tau_\gamma$, $\tau_4 = \tau_{\theta_A}$, $\tau_5 = \tau_{\theta_B}$, $\tau_6 = \tau_{\theta_C}$.

τ_α, τ_β, τ_γ describe the components reaction torque impacted on cube along the axes of body frame:

$$\begin{bmatrix}\tau_\alpha\\ \tau_\beta\\ \tau_\gamma\end{bmatrix} = -I_w \cdot T_{bf}\begin{bmatrix}\ddot{\theta}_A\\ \ddot{\theta}_B\\ \ddot{\theta}_C\end{bmatrix} \tag{8}$$

τ_{θ_A}, τ_{θ_B}, τ_{θ_C} are the components of Coriolis inertia force impacted on reaction wheels along the axes of reaction wheels frame:

$$\begin{bmatrix}\tau_{\theta_A}\\ \tau_{\theta_B}\\ \tau_{\theta_C}\end{bmatrix} = \begin{bmatrix}\tau_A\\ \tau_B\\ \tau_C\end{bmatrix} - I_w \cdot T_{bf}^T\begin{bmatrix}\ddot{\alpha}\\ \ddot{\beta}\\ \ddot{\gamma}\end{bmatrix} \tag{9}$$

E. Dynamic System Derivation

We can put (1)-(5) into (7) to get this:

$$\begin{cases} I_y\ddot{\alpha} = mgl\sin\alpha\cos\beta + \tau_\alpha \\ I_x\ddot{\beta} = mgl\sin\beta\cos\alpha + \tau_\beta \\ I_z\ddot{\gamma} = \tau_\gamma \\ I_w\ddot{\theta}_A = \tau_{\theta_A} \\ I_w\ddot{\theta}_B = \tau_{\theta_B} \\ I_w\ddot{\theta}_C = \tau_{\theta_C} \end{cases}$$

Substitute τ_α, τ_β, τ_γ and τ_{θ_A}, τ_{θ_B}, τ_{θ_C} in (8) and (9) with τ_A, τ_B and τ_C and use (6) to use input voltage to represent torque from the motor.

After simplification, we substitute the θ in first three equations with the last three equations.

The result is like this:

$$
\begin{cases}
(I_y - I_w)\ddot{\alpha} & = mgl\sin\alpha\cos\beta - \frac{\sqrt{6}}{6}\frac{K_t}{R_m}[(2v_A - v_B - v_C) \\
& \quad -K_t(2\dot{\theta}_A - \dot{\theta}_B - \dot{\theta}_C)] \\
(I_x - I_w)\ddot{\beta} & = mgl\sin\beta\cos\alpha - \frac{\sqrt{2}}{2}\frac{K_t}{R_m}[(v_B - v_C) \\
& \quad -K_t(\dot{\theta}_B - \dot{\theta}_C)] \\
(I_z - I_w)\ddot{\gamma} & = -\frac{\sqrt{3}}{3}\frac{K_t}{R_m}[(v_A + v_B + v_C) - \\
& \quad K_t(\dot{\theta}_A + \dot{\theta}_B + \dot{\theta}_C)] \\
I_w\ddot{\theta}_A & = K_1 v_A + K_2 v_B + K_3 v_C \\
& \quad -K_t(K_1\dot{\theta}_A + K_2\dot{\theta}_B + K_3\dot{\theta}_C) \\
I_w\ddot{\theta}_B & = K_4 v_A + K_5 v_B + K_6 v_C \\
& \quad -K_t(K_4\dot{\theta}_A + K_5\dot{\theta}_B + K_6\dot{\theta}_C) \\
I_w\ddot{\theta}_C & = K_7 v_A + K_8 v_B + K_9 v_C \\
& \quad -K_t(K_7\dot{\theta}_A + K_8\dot{\theta}_B + K_9\dot{\theta}_C)
\end{cases}
$$

where the K_i is as follows:

$$K_1 = \frac{K_t}{R_m}(\frac{1}{I_w} + \frac{2}{3(I_y - I_w)} + \frac{1}{3(I_z - I_w)})$$

$$K_2 = K_3 = K_4 = K_7 = \frac{K_t}{R_m}(-\frac{1}{3(I_y - I_w)} + \frac{1}{3(I_z - I_w)})$$

$$K_5 = \frac{K_t}{R_m}(\frac{1}{I_w} + \frac{1}{6(I_y - I_w)} + \frac{1}{2(I_x - I_w)} + \frac{1}{3(I_z - I_w)})$$

$$K_6 = K_8 = \frac{K_t}{R_m}(\frac{1}{6(I_y - I_w)} - \frac{1}{2(I_x - I_w)} + \frac{1}{3(I_z - I_w)})$$

$$K_9 = \frac{K_t}{R_m}(\frac{1}{I_w} + \frac{1}{6(I_y - I_w)} + \frac{1}{2(I_x - I_w)} + \frac{1}{3(I_z - I_w)})$$

Set our state vector $x = [\dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \alpha\ \beta\ \dot{\theta}_A\ \dot{\theta}_B\ \dot{\theta}_C]^T$ and our input vector $u = [v_A\ v_B\ v_C]^T$.

Set the output vector like $y = [\dot{\gamma}\ \alpha\ \beta\ \dot{\theta}_A\ \dot{\theta}_B\ \dot{\theta}_C]^T$ and the input vector $u = [v_a \quad v_b \quad v_c]$

The state space equation of this system is

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

F. Simplification of the Model

From previous part, we used Coriolis Force to derive the system. However, since we cannot control the yaw angle γ, the angle velocity of γ could be a very big value. The reason of this problem is that we need some big torque at the very beginning and this could gives a very fast constant angular velocity to the yaw angle which will lead to the explosion of Coriolis Force.

For the simplification, we can remove the Coriolis Force the system impacting on the reaction wheels to prevent explosion. Also, we can save the computation of those complicated paramters $(K_1, K_2, ...)$ And the new system is shown below:

$$
\begin{cases}
(I_y - I_w)\ddot{\alpha} & = mgl\sin\alpha\cos\beta - \frac{\sqrt{6}}{6}\frac{K_t}{R_m}[(2v_A - v_B - v_C) \\
& \quad -K_t(2\dot{\theta}_A - \dot{\theta}_B - \dot{\theta}_C)] \\
(I_x - I_w)\ddot{\beta} & = mgl\sin\beta\cos\alpha - \frac{\sqrt{2}}{2}\frac{K_t}{R_m}[(v_B - v_C) \\
& \quad -K_t(\dot{\theta}_B - \dot{\theta}_C)] \\
(I_z - I_w)\ddot{\gamma} & = -\frac{\sqrt{3}}{3}\frac{K_t}{R_m}[(v_A + v_B + v_C) - \\
& \quad K_t(\dot{\theta}_A + \dot{\theta}_B + \dot{\theta}_C)] \\
I_w\ddot{\theta}_A & = \frac{K_t}{R_m}(v_A - K_t\dot{\theta}_A) \\
I_w\ddot{\theta}_B & = \frac{K_t}{R_m}(v_B - K_t\dot{\theta}_B) \\
I_w\ddot{\theta}_C & = \frac{K_t}{R_m}(v_C - K_t\dot{\theta}_C)
\end{cases}
$$

G. State Space Representation

After the above operations we can get our state space model

$$
A = \begin{bmatrix}
0 & 0 & 0 & \frac{mgl}{I_y - I_w} & 0 & \frac{\sqrt{6}K_t^2}{3R_m(I_y - I_w)} & -\frac{\sqrt{6}K_t^2}{3R_m(I_y - I_w)} & -\frac{\sqrt{6}K_t^2}{3R_m(I_y - I_w)} \\
0 & 0 & 0 & 0 & \frac{mgl}{I_x - I_w} & 0 & \frac{\sqrt{2}K_t^2}{2R_m(I_x - I_w)} & -\frac{\sqrt{2}K_t^2}{2R_m(I_x - I_w)} \\
0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{3}K_t^2}{3R_m(I_z - I_w)} & -\frac{\sqrt{3}K_t^2}{3R_m(I_z - I_w)} & -\frac{\sqrt{3}K_t^2}{3R_m(I_z - I_w)} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{-K_t^2}{R_m I_w} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{-K_t^2}{R_m I_w} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-K_t^2}{R_m I_w}
\end{bmatrix}
$$

$$
B = \begin{bmatrix}
-\frac{\sqrt{6}K_t}{3R_m(I_y - I_w)} & \frac{\sqrt{6}K_t}{6R_m(I_y - I_w)} & \frac{\sqrt{6}K_t}{6R_m(I_y - I_w)} \\
0 & -\frac{\sqrt{2}K_t}{2R_m(I_x - I_w)} & \frac{\sqrt{2}K_t}{2R_m(I_x - I_w)} \\
-\frac{\sqrt{3}K_t}{3R_m(I_z - I_w)} & -\frac{\sqrt{3}K_t}{3R_m(I_z - I_w)} & -\frac{\sqrt{3}K_t}{3R_m(I_z - I_w)} \\
0 & 0 & 0; \\
0 & 0 & 0; \\
\frac{K_t}{R_m I_w} & 0 & 0 \\
0 & \frac{K_t}{R_m I_w} & 0 \\
0 & 0 & \frac{K_t}{R_m I_w}
\end{bmatrix}
$$

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

$$D = 0$$

TABLE I
CONSTANT PARAMETERS

Table	Constant Values		
I_x	0.014484	I_y	0.014484
I_z	0.004026	I_w	0.000189
L	0.108	m	1.146
K_t	0.0251	R_m	0.0464
g	9.81		

IV. Results

A. Controllability and Observability

The constants in the system are shown in Table I, we substitute these constants to get our system (A, B, C, D).

We can derive the eigenvalue of A and we can see the eigenvalues are real but do not always have negative real part. This means my system is not internally stable.

The controllability matrix

$$C = [B\ AB\ A^2B\ A^3B\ A^4B\ A^5B\ A^6B\ A^7B]$$

has $rank = 8$ which means this system is **controllable**.

My model is a dynamic system and we can directly set the output to be the states. To test the Luenberger observer, instead of setting the output to be directly states, we reduced two states but still make the system observable. This is how be desgigned the C matrix. The observability matrix

$$C = [C\ CA\ CA^2\ CA^3\ CA^4\ CA^5\ CA^6\ CA^7]^T$$

has $rank = 8$ which means this system is **observable**.

B. Simulation

This part is to test if this system is consistent with the reality.

a) Zero Input Response: This simulation start with $20°$ at α angle and $0°$ at β angle. We did the simulation in 0.5 second. From Fig. 2, we can see the reaction wheel is spinning with rotation axis x due to we have no limitation for α.

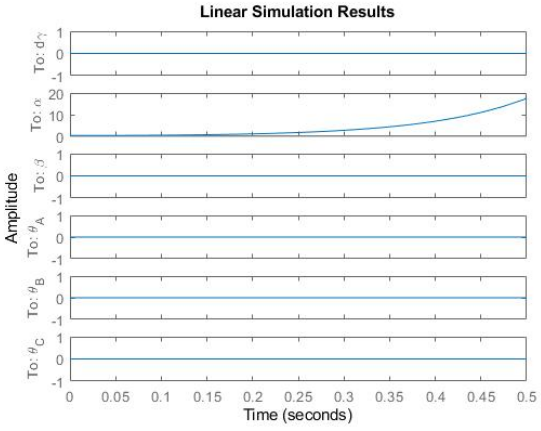

Fig. 2. Simulation with zero input and $\alpha = 20°$

b) Zero State Response: The zero state response start with zero state and has a input of $[0; 0.5; -0.5]^T$ at all the time. From Fig. 3, we can see the reaction wheel A keep static while B and C are rotating in different directions while the speed is around 20 rad/s This is consistent with our settings.

From the result above we can get that our model is consistent with the reality.

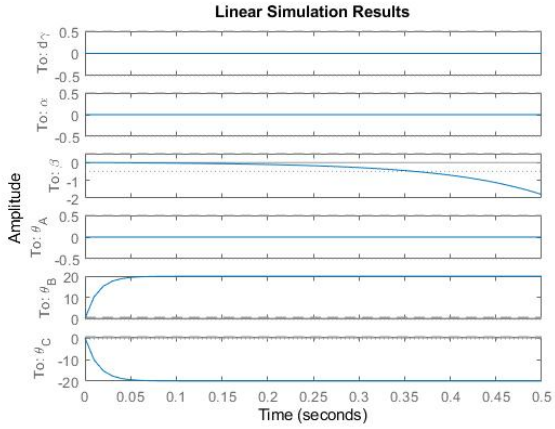

Fig. 3. Simulation with zero state and constant input $[0; 0.5; -0.5]^T$

C. LQR

Since we already have the state space model, we can use LQR controller as a initial trial. The algorithm of continuous LQR is

$$J(u) = \int_0^\infty x^T Q x + u^T R u\, dt$$

The most important part is to design the Q and R matrix. As we can see the Q matrix is used to penalize the state while R is used to penalize the input. If we want to make the state converge faster, we need to set Q larger than R and vice versa. Since what we are trying to control is the α and β angle, which are the states, we need to make Q larger than R. First I choose $Q = diag(100, 100, 100, 100, 100, 100, 100, 100)$ and $R = diag(0.01, 0.01, 0.01)$. We got the control result in Fig. 4. The controller we designed just meet our needs. As we can see from the figure, there's a peak value in $\dot{\theta}_c$, which is the angular velocity of c reaction wheel. The peak value is around 600 `rad/s`, which is also 5400 `rpm`. Although it seems very high but it is totally acceptable for a BLDC motor in our daily life. Also, for the result of the controller, we can see that all the states converges in 1 second. This means the controller is very practical for our cube robot.

D. Luenberger Observer

In practice we cannot see the inner state or it is hard to measure some states. That's why we need the observer to see the inner state. We learned Luenberger observer in class. The characteristic of Luenberger observer is that the state w is the estimate $\hat{x}$. Since my system is a dynamic system, the output is just simply the states. So I erased some states in the output so that I can design an observer for this system.

We need to design an L to make sure $A + LC$ is stable by solving its LMI problem:

$$\begin{aligned} W & \succ 0 \\ A^T W + WA + C^T V + V^T C & \prec 0 \end{aligned}$$

where $L = W^{-1}V^T$.

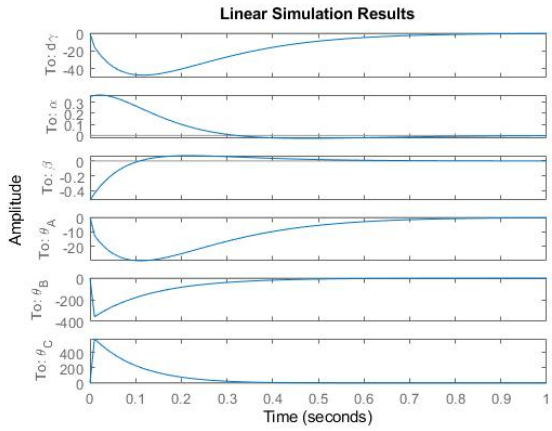

Fig. 4. Simulation with LQR control

After solving the LMI problem with `cvx`, we get the L we want. According to the Separation Principle, we can design the observer and controller separately. Here I'm going to use the controller we get in LQR to do to simulation.

The combinitional system of the observer and controller has the result in Fig. 5. We can see the observer almost follows the original state with some initial conditions.

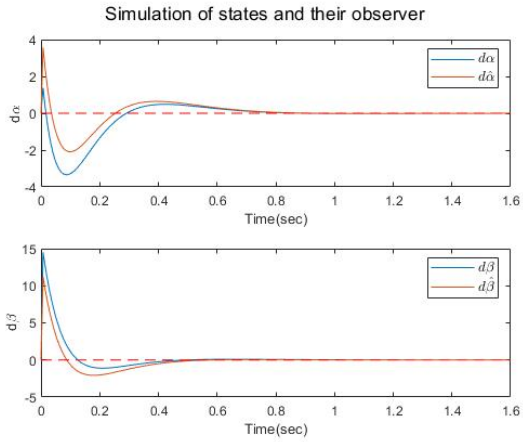

Fig. 5. Luenberger observer with LQR controller

E. $\mathcal{H}_2$ Optimal Control

Consider the full information control problem here. Assume the exogenous input of the closed-loop system is some disturbance applied to the system. Let's say we have some disturbance which caused $B_1 = 0.1B_2$. Also since our system is dynamic, we can just make $C = I$, this means the output is just the state. Therefore, the system of G is

$$G = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline I & 0 & 0 \\ I & 0 & 0 \end{array}\right]$$

$$K = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & F \end{array}\right]$$

The closed-loop system is

$$\mathcal{F}_l(G,K) = \left[\begin{array}{c|c} A+B_2F & B_1 \\ \hline C_1+D_{12}F & D_{11} \end{array}\right] = \left[\begin{array}{c|c} A+B_2F & B_1 \\ \hline I & 0 \end{array}\right]$$

The feedback controller F can be derived by solving the LMI problem

$$\begin{aligned} \min \quad & \gamma^2 \\ \text{s.t.} \quad & trace(W) < \gamma^2 \\ & [A \quad B_2]\begin{bmatrix} X \\ Z \end{bmatrix} + [X \quad Z^T]\begin{bmatrix} A^T \\ B_2^T \end{bmatrix} + B_1B_1^T \prec 0, \\ & X \succ 0 \\ & \begin{bmatrix} X & (C_1X + D_{12}Z)^T \\ (C_1X + D_{12}Z) & W \end{bmatrix} \succ 0 \end{aligned}$$

where $F = ZX^{-1}$.

We solve F with `cvx` and then do the simulation for this closed loop system. The initial condition response is shown in Fig. 6. Since we are trying to mark w as a disturbance, I use some random inputs which follow normal distribution $N(0,1)$ and the result is shown in Fig. 7. The grey line is the disturbance input. The controller shows a very impressive result. Although we have some disturbance, the states converge as we thought.

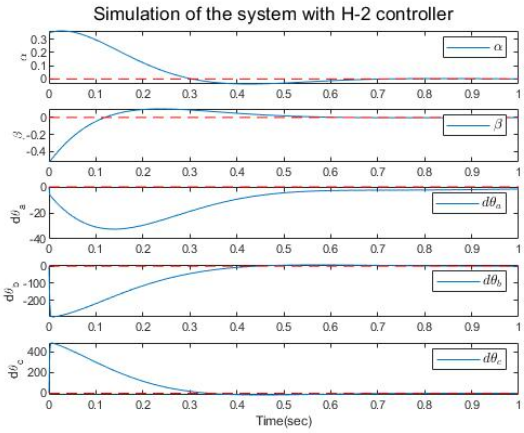

Fig. 6. Initial condition response of $\mathcal{H}_2$ optimal control

We also want to make sure the controller input is acceptable in reality. So we checked the control input here.

```
>> max(abs(F*x_h2'),[],2)
ans =
  126.8252
  314.6543
  900.1839
```

.

From the result shown above, the maximum value of input is extremely high and it is up to 900 volts. This is not acceptable

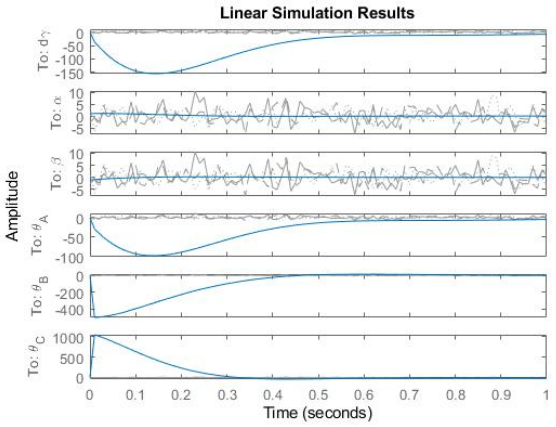

Fig. 7. Simulation of $\mathcal{H}_2$ optimal control with disturbance

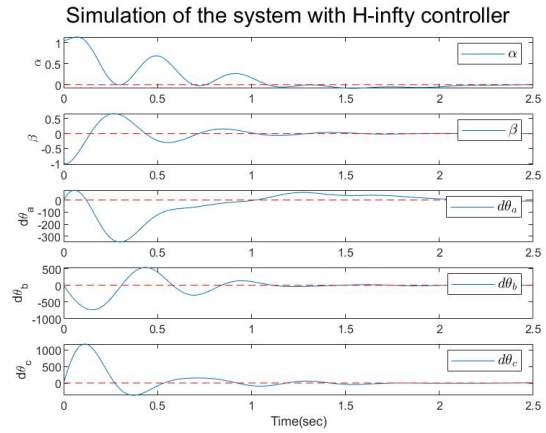

Fig. 8. Simulation of $\mathcal{H}_\infty$ optimal control with initial state

in reality because we cannot set such a high voltage for our BLDC motor.

F. $\mathcal{H}_\infty$ State Feedback Control

As we know, the $\mathcal{H}_2$ optimal control is doing control over minimizing the average "energy" of the closed-loop system while $\mathcal{H}_\infty$ minimizes the maximum "energy" of the closed-loop system.

Here we can assume we have the same closed loop system as the $\mathcal{H}_2$ control (IV-E). The goal of the controller design is to minimize $\|\mathcal{F}_l(G,K)\|_{\mathcal{H}_\infty}$. So we can use the KYP Lemma to design the LMI question which is equivalent to this goal. Here we choose the dual KYP conditions. The LMI question which is also a Semi-definite Program can be put like this:

$$\begin{aligned} \min_{\gamma,Y} \quad & \gamma \\ \text{s.t.} \quad & \begin{bmatrix} YA^T + AY + Z^T B_2^T + B_2 Z & B_1 & YC_1 + Z^T D_{12}^T & 0 \\ B_1^T & -\gamma I & D_{11}^T & 0 \\ C_1 Y + D_{12} Z & D_{11} & -\gamma I & 0 \\ 0 & 0 & 0 & -Y \end{bmatrix} \prec 0 \end{aligned}$$

After solving this LMI problem we can get the controller $F = ZY^{-1}$. The initial condition response is shown in Fig. 8 and the response with white noise disturbance is shown in Fig. 9.

In Fig. 8, we can see the overhead of the signals are much bigger than the $\mathcal{H}_2$ signal in Fig. 6. And the converge time is longer. As disturbance, some random noise are added to the exogenous input w. However, we can see the controller is very robust with small noise since the states converges even with disturbance.

Also, we checked the input with this controller to see if the input value is reasonable. In our full information control problem we know that the feedback loop has $u = Fx$. The control input of the system is just the state times the controller. Therefore, we can search for the maximum value of the control input. For the controller I designed, we can see the maximum input is like

```
>> max(abs(F*x_hinf'),[],2)
ans =
    8.8259
   18.6715
   30.1884
```

.

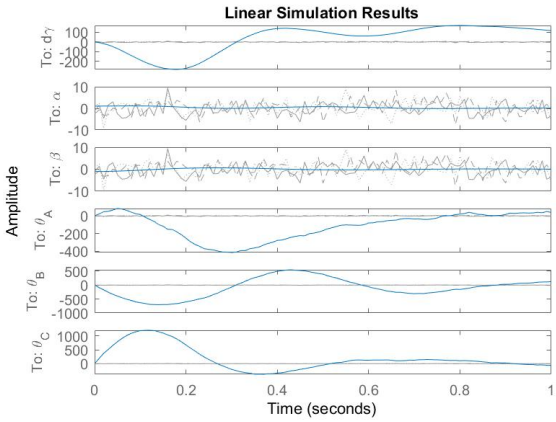

Fig. 9. Simulation of $\mathcal{H}_i nfty$ optimal control with disturbance

These are all reasonable since our input is the armature voltage of the motor. The maximum voltage is around 30 volts. This is acceptable. So this controller is practical for the system.

G. Controller Tuning

The controller we learned this semester is only about how to design a controller from 0 to 1 instead of how to make the controller better. Therefore, in my controller designing process, the overhead and converging time are not considered.

For the controller tuning, I simply restricted the conditions furthermore in LMI to tune the controller. For example, in the LMI problem of $\mathcal{H}_2$, we only have conditions such as less than 0 or bigger than 0. But I used negative value for a less than sign instead of a true zero. I used this method because the `cvx` will only optimize the conditions right to the equality in LMI. The zero in the inequality is just a relaxation of existance of the controller.

Conclusion and Potential Future Work

In this project, we derived a model for the cubic robot which can stand on its corner. And then we derived some different controllers for this dynamic model such as LQR, $\mathcal{H}_2$ optimal control, $\mathcal{H}_\infty$ state feedback control. In addition to that, Luenberger observer is also implemented with the LQR controller.

In addition, I used the model from the paper **??**. But actually their paper have many problems and I did the calculation again to fix the model. This takes me a lot of time but I think it's meaningful.

However, I do not have much experience to tune the model with pole placement or the root locus process. So in this project only the basic tuning are deployed and some of them may not have an ideal effect. I'm very interested in this model even though it is a higher level version of inverted pendulum.

Therefore, in the future, I may still have some future plan for this model. I did not include my MPC control in this project due to time limitation because whatever the horizon I set for the MPC it just blew up. It must be my reason and I think in the future I can handle it with more experience and knowledge.

References

[1] M. Gajamohan, M. Muehlebach, T. Widmer and R. D'Andrea, "The Cubli: A reaction wheel based 3D inverted pendulum," 2013 European Control Conference (ECC), Zurich, Switzerland, 2013, pp. 268-274, doi: 10.23919/ECC.2013.6669562.

[2] Q. Yin, F. Wang, S. Lu, Y. Fan, Y. Liu and G. Li, "Research on Balance Control of Cube Robot," 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR), Xining, China, 2021, pp. 462-467, doi: 10.1109/RCAR52367.2021.9517632.

[3] Z. Chen, X. Ruan and Y. Li, "Balancing control of a cubical robot balancing on its corner," 2018 IEEE 15th International Workshop on Advanced Motion Control (AMC), Tokyo, Japan, 2018, pp. 631-636, doi: 10.1109/AMC.2019.8371167.

```
%% Load Parameters
%Inertia: kg*m^2
clear;
Ix = 0.014484;
Iy = Ix;
Iz = 0.004026;
Iw = 0.000189;
%Length: m
L = 0.108;
%mass: kg
m = 1.146;
%Moment constant of motor: N*m/A
Kt = 0.0251;
%Resistor: olm
Rm = 0.0464;
%Gravity constant: m/s^2
g = 9.81;


%% System Description
modelSelect = 0;
%0 means simplified model
%1 means original model.

if modelSelect == 0
    A = [0 0 0 m*g*L/(Iy-Iw) 0             sqrt(6).*Kt.^2/(3*Rm.*(Iy-Iw))  -sqrt(6).*
        Kt.^2/(3*Rm.*(Iy-Iw)) -sqrt(6).*Kt.^2/(3*Rm.*(Iy-Iw));
          0 0 0 0             m*g*L/(Ix-Iw) 0                                sqrt(2).*
             Kt.^2/(2*Rm.*(Ix-Iw))  -sqrt(2).*Kt.^2/(2*Rm.*(Ix-Iw));
          0 0 0 0             0             -sqrt(3).*Kt.^2/(3*Rm.*(Iz-Iw)) -sqrt(3).*
             Kt.^2/(3*Rm.*(Iz-Iw)) -sqrt(3).*Kt.^2/(3*Rm.*(Iz-Iw));
          1 0 0 0             0             0                                0
                                           0                              ;
          0 1 0 0             0             0                                0
                                           0                              ;
          0 0 0 0             0             -Kt.^2/(Iw.*Rm)                  0
                                           0                              ;
          0 0 0 0             0             0                                -Kt.^2/(Iw
             .*Rm)                  0                                ;
          0 0 0 0             0             0                                0
                                           -Kt.^2/(Iw.*Rm)                ];

    B = [-sqrt(6).*Kt/(3.*Rm.*(Iy-Iw)) sqrt(6).*Kt/(6.*Rm.*(Iy-Iw))    sqrt(6).*Kt
        /(6.*Rm.*(Iy-Iw)) ;
          0                            -sqrt(2).*Kt/(2.*Rm.*(Ix-Iw))   sqrt(2).*Kt
             /(2.*Rm.*(Ix-Iw)) ;
          -sqrt(3).*Kt/(3.*Rm.*(Iz-Iw)) -sqrt(3).*Kt/(3.*Rm.*(Iz-Iw))   -sqrt(3).*Kt
             /(3.*Rm.*(Iz-Iw));
          0                            0                                0
                                       ;
          0                            0                                0
                                       ;
          Kt/(Rm.*Iw)                  0                                0
                                       ;
```

```
         0                                Kt/(Rm.*Iw)                          0
                                          ;
         0                                0                                    Kt/(Rm.*Iw)
                              ];
else
    %dtheta A
    K1 = Kt/Rm*(1/Iw+ 2/(3*(Iy-Iw)) + 1/(3*(Iz-Iw)));
    K2 = Kt/Rm*( -1/(3*(Iy-Iw)) + 1/(3*(Iz-Iw)));
    K3 = Kt/Rm*( -1/(3*(Iy-Iw)) + 1/(3*(Iz-Iw)));

    %dtheta B
    K4 = Kt/Rm*(-1/(3*(Iy-Iw)) + 1/(3*(Iz-Iw)));
    K5 = Kt/Rm*(1/Iw + 1/(6*(Iy-Iw)) + 1/(2*(Ix-Iw)) + 1/(3*(Iz-Iw)));
    K6 = Kt/Rm*(1/(6*(Iy-Iw)) - 1/(2*(Ix-Iw)) + 1/(3*(Iz-Iw)));

    %dtheta C
    K7 = Kt/Rm*(-1/(3*(Iy-Iw)) + 1/(3*(Iz-Iw)));
    K8 = Kt/Rm*(1/(6*(Iy-Iw)) - 1/(2*(Ix-Iw)) + 1/(3*(Iz-Iw)));
    K9 = Kt/Rm*(1/Iw + 1/(6*(Iy-Iw)) + 1/(2*(Ix-Iw)) + 1/(3*(Iz-Iw)));


     A = [0 0 0 m*g*L/(Iy-Iw)             0                              sqrt(6).*Kt
         .^2/(3*Rm.*(Iy-Iw))      -sqrt(6).*Kt.^2/(3*Rm.*(Iy-Iw))    -sqrt(6).*Kt.^2/(3*
         Rm.*(Iy-Iw));
          0 0 0 0                         m*g*L/(Ix-Iw)                0
                                               sqrt(2).*Kt.^2/(2*Rm.*(Ix-Iw))    -sqrt
              (2).*Kt.^2/(2*Rm.*(Ix-Iw));
          0 0 0 0                         0                              -sqrt(3).*Kt
              .^2/(3*Rm.*(Iz-Iw))     -sqrt(3).*Kt.^2/(3*Rm.*(Iz-Iw))    -sqrt(3).*Kt
              .^2/(3*Rm.*(Iz-Iw));
          1 0 0 0                         0                              0
                                               0                                   0
                                          ;
          0 1 0 0                         0                              0
                                               0                                   0
                                          ;
          0 0 0 m*g*L*sqrt(6)/(3*(Iw-Iy)) 0                              -Kt*K1
                                         -Kt*K2                                -Kt*K3
                                    ;
          0 0 0 m*g*L*sqrt(6)/(6*(Iy-Iw)) -m*g*L*sqrt(2)/(2*(Ix-Iw))   -Kt*K4
                                         -Kt*K5                                -Kt*K6
                                    ;%
          0 0 0 m*g*L*sqrt(6)/(6*(Iy-Iw)) m*g*L*sqrt(2)/(2*(Ix-Iw)) -Kt*K7
                                         -Kt*K8                                -Kt*K9
                                    ];%

    B = [-sqrt(6).*Kt/(3.*Rm.*(Iy-Iw)) sqrt(6).*Kt/(6.*Rm.*(Iy-Iw))    sqrt(6).*Kt
        /(6.*Rm.*(Iy-Iw)) ;
          0                              -sqrt(2).*Kt/(2.*Rm.*(Ix-Iw))    sqrt(2).*Kt
              /(2.*Rm.*(Ix-Iw)) ;
          -sqrt(3).*Kt/(3.*Rm.*(Iz-Iw)) -sqrt(3).*Kt/(3.*Rm.*(Iz-Iw))    -sqrt(3).*Kt
              /(3.*Rm.*(Iz-Iw));
          0                              0                                   0
                                         ;
          0                              0                                   0
                                         ;
```

```
        K1                              K2                              K3
                                       ;
        K4                              K5                              K6
                                       ;%
        K7                              K8                              K9
                                       ];%

end
%C = [zeros(5,3) diag(ones(5,1))];
C =[0 0 1 0 0 0 0 0;
    0 0 0 1 0 0 0 0;
    0 0 0 0 1 0 0 0;
    0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 1];

D=0;

%% Check Controllability
rank(ctrb(A,B))

%% Check Observability
rank(obsv(A,C))
%% Zero Input Simulation
states = {'d\alpha','d\beta','d\gamma','\alpha','\beta','\theta_A','\theta_B','\
   theta_C'};
inputs = {'v_a','v_b','v_c'};
outputs = {'d\gamma','\alpha','\beta','\theta_A','\theta_B','\theta_C'};
T = 0:0.01:0.5; %Simulate for 0.5 second

%Zero Input Simulation
U = zeros(3,length(T));
x0 = [0 0 0 20/180*pi 0 0 0 0]; %20 degree
sys = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
lsim(sys,U,T,x0)


%% Zero State Simulation
U = [zeros(1,length(T));0.5*ones(1,length(T));-0.5*ones(1,length(T))];
%U = [0.5*ones(1,length(T));-0.25*ones(1,length(T));-0.25*ones(1,length(T))];
%U = [0.25*ones(1,length(T));0.25*ones(1,length(T));0.25*ones(1,length(T))];


%initial condition
x0 = [0 0 0 0 0 0 0 0];

sys = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
lsim(sys,U,T,x0)

%impulse(sys)
%% LQR control
Q = diag(100*ones(1,8));
R = diag(0.01*ones(1,3));
[K,S,P] = lqr(sys,Q,R);
T = 0:0.01:1; %Simulate for 1 second
U = [zeros(3,length(T))];
```

```
sys2 = ss(A-B*K,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
x0 = [0 0 0 20/180*pi -30/180*pi 0 0 0];
lsim(sys2,U,T,x0)
```

```
%% Luenberger Observer

cvx_begin sdp
    variable W(8,8) symmetric
    variable V(6,8)
    LMI = A'*W+W*A+C'*V+V'*C;
    W >= 0.01*eye(8);
    LMI <= -10000*eye(8);
cvx_end

L = W\V';

%Compute eigenvalue of A+LC to determine the stabiblity
eig(A+L*C)

%Observer Simulation
A_ob = [A -B*K; -L*C A+L*C-B*K];
B_ob = [zeros(8,3);zeros(8,3)];
C_ob = eye(16); %Set output to be the states
D_ob = 0;

sys_ob = ss(A_ob,B_ob,C_ob,D_ob);

x0 = [0 0 0 20/180*pi -30/180*pi 0 0 0 zeros(1,8)];
[y_cl,t,x_cl] = initial(sys_ob,x0);

%Plot
subplot(2,1,1)
plot(t,x_cl(:,1),t,x_cl(:,9),t,zeros(size(t)),'r--')
set(legend('$$d\alpha$$','$$d\hat{\alpha}$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1.6])
xlabel('Time(sec)')
ylabel('d\alpha')

subplot(2,1,2)
plot(t,x_cl(:,2),t,x_cl(:,10),t,zeros(size(t)),'r--')
set(legend('$$d\beta$$','$$d\hat{\beta}$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1.6])
xlabel('Time(sec)')
ylabel('d\beta')

sgtitle('Simulation of states and their observer')
```

```
%% H-2 Optimal Control
B1 = 0.1.*B;
cvx_begin sdp
    variable X(8,8) symmetric
    variable Z(3,8)
    variable W(8,8) symmetric
    variable gsq
    minimize ( gsq )
```

```
    [A B]*[X; Z]+[X Z']*[A'; B']+ B1*B1' <= -100*eye(8);
    [X   X';
     X   W] >= 1*eye(16);
    trace (W) <= gsq ;
    X >= 0;
cvx_end

F = Z/X;

sysh2 = ss(A+B*F,B1,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
x0 = [0 0 0 60/180*pi -60/180*pi 0 0 0];

%T = 0:0.01:1; %Simulate for 1 second
%U = [3*randn(3,length(T))];
%lsim(sysh2,U,T,x0)
[y_h2,t,x_h2] = initial(sysh2,x0);

%Plot
subplot(5,1,1)
plot(t,x_h2(:,4),t,zeros(size(t)),'r--')
set(legend('$$\alpha$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1])

ylabel('\alpha')

subplot(5,1,2)
plot(t,x_h2(:,5),t,zeros(size(t)),'r--')
set(legend('$$\beta$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1])

ylabel('\beta')

subplot(5,1,3)
plot(t,x_h2(:,6),t,zeros(size(t)),'r--')
set(legend('$$d\theta_a$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1])

ylabel('d\theta_a')

subplot(5,1,4)
plot(t,x_h2(:,7),t,zeros(size(t)),'r--')
set(legend('$$d\theta_b$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1])

ylabel('d\theta_b')

subplot(5,1,5)
plot(t,x_h2(:,8),t,zeros(size(t)),'r--')
set(legend('$$d\theta_c$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,1])
xlabel('Time(sec)')
ylabel('d\theta_c')


sgtitle('Simulation of the system with H-2 controller')
```

```
%% H2 with disturbance
%Add disturbance to the input
T = 0:0.01:1; %Simulate for 1 second
U = [3*randn(3,length(T))];
lsim(sysh2,U,T,x0)

max(abs(F*x_h2'),[],2)
```

```
%% H-infty
cvx_begin sdp
    variable Y(8,8) symmetric
    variable gam
    variable Z(3,8)
    min gam
    [Y*A'+A*Y+Z'*B'+B*Z,   B1,               Y,              zeros(8,8);
     B1',                  -gam*eye(3),     zeros(3,8),     zeros(3,8);
     Y,                    zeros(8,3),      -gam*eye(8),    zeros(8,8);
     zeros(8,8),           zeros(8,3),     zeros(8,8),      -Y]<=0;
cvx_end

F = Z/Y;

syshinf = ss(A+B*F,B1,C,D,'statename',states,'inputname',inputs,'outputname',outputs)
    ;
x0 = [0 0 0 60/180*pi -60/180*pi 0 0 0];

%T = 0:0.01:1; %Simulate for 1 second
%U = [3*randn(3,length(T))];
%lsim(sysh2,U,T,x0)
[y_hinf,t,x_hinf] = initial(syshinf,x0);

%Plot
subplot(5,1,1)
plot(t,x_hinf(:,4),t,zeros(size(t)),'r--')
set(legend('$$\alpha$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,2.5])

ylabel('\alpha')

subplot(5,1,2)
plot(t,x_hinf(:,5),t,zeros(size(t)),'r--')
set(legend('$$\beta$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,2.5])

ylabel('\beta')

subplot(5,1,3)
plot(t,x_hinf(:,6),t,zeros(size(t)),'r--')
set(legend('$$d\theta_a$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,2.5])

ylabel('d\theta_a')

subplot(5,1,4)
plot(t,x_hinf(:,7),t,zeros(size(t)),'r--')
set(legend('$$d\theta_b$$'),'Interpreter','Latex','FontSize', 10);
```

```
xlim([0,2.5])

ylabel('d\theta_b')

subplot(5,1,5)
plot(t,x_hinf(:,8),t,zeros(size(t)),'r--')
set(legend('$$d\theta_c$$'),'Interpreter','Latex','FontSize', 10);
xlim([0,2.5])
xlabel('Time(sec)')
ylabel('d\theta_c')


sgtitle('Simulation of the system with H-infty controller')
%% H infty disturbance
%Add disturbance to the input
T = 0:0.01:1; %Simulate for 1 second
U = [3*randn(3,length(T))];
lsim(syshinf,U,T,x0)

max(abs(F*x_hinf'),[],2)
```