MAE 204
Winter 2021
FINAL PROJECT

YA-CHE SHIH

Contents

Milestone 1: youBot Kinematics Simulator and csv Output

Nextstate function

Function:

6 Inputs:

- Configuration of chassis: *3 values*
  - φ (chassis face direction w.r.t space coordinate), x, y
- Configuration of arms: *5 values*
  - $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ (5 theta represent angle of each arm)
- Angle of wheels: *4 values*
  - w_1, w_2, w_3, w_4 (4 omegas represent angle of each wheel)
- Arms speed: 5 values
  - $d\theta_1, d\theta_2, d\theta_3, d\theta_4, d\theta_5$(5 theta dot=speed of each arm)
- Wheels speed: *4 values*
  - u_1, u_2, u_3, u_4 (4 u represent speed of each wheel)
- Time step: *1 value*
  - dt (In this project we set dt=0.01s)

1 Output:

- Next configuration: *12 values*
  - φ, x, y, $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, w_1, w_2, w_3, w_4$

Approach:

1. Set maximum velocity of wheels and arms speed. If velocity exceeds maximum value, it can only be maximum value.
2. Apply F in Modern Robotic to find body twist V_b6, and then find transformation matrix from current b to next b' ($T_{bb'}$).
3. Take inputs φ, x, y to find T_{sb}, and then we can obtain $T_{sb'}$.
4. Find next chassis configuration by $T_{sb'}$. (First, I tried to find new φ by arccos(1/2*(trace(R)-1), but by this approach $\varphi \in [0,\pi]$, . One day I realized since chassis can only rotate along z-axis, I can simply find φ by arcsin(R(2,1)).
5. Next arms configuration=current configuration (θ)+ speed ($d\theta$)
   Next wheels angle=current angle(w) + speed(u)
6. Output nextstate

Milestone 2: **Reference Trajectory Generation**

Trajectory generator

Function:

4 Inputs:

- Initial end-effector configuration: *1 matrix*

  T_{se} (transformation matrix from space to end-effector)
- Cube's initial position: *1 matrix*

  $T_{sc,initial}$ (transformation matrix from space to cube's initial position)
- Cube's final position: *1 matrix*

  $T_{sc,final}$ (transformation matrix from space to cube's final position)
- Time step: *1 value*

  dt (In this project we set dt=0.01s).

3 Output:

- Trajectory matrices:

  generate trajectory transformation matrix (T_{se}) from space to end-effector
- gripper close time & open time:

  save the time when gripper close and open

Approach:

1. Set grasp angle between gripper and cube. (150°)
2. Find transformation matrix of standoff_initial, grasp_initial, standoff_final, grasp_final.
3. Set moving time and trajectory mothod.
4. I set k=10, which means in N second, we can obtain N*k/dt trajectory matrixes.
5. Assign gripper closing time and open time.
6. Output trajectory matrices, gripper closing and open time.

Milestone 3: Feedforward Control

Feedback control

Function:

7 Input:

- Chassis configuration: *3 values*
  - φ (chassis face direction w.r.t space coordinate), x, y
- Desired configuration: *1 matrix*
  - X_{desire} (obtained from trajectory matrices)
- Next desired configuration: *1 matrix*
  - $X_{desire,next}$ (obtained from trajectory matrcies)
- Kp: *6x6 matrix*
- Ki: *6x6 matrix*
- Time step: *1 value*
  - dt (In this project we set dt=0.01s).
- Added error twist in former time:
  - X_{error} will accumulate and added in twist.

4 Output:

- Arms speed: *5 values*
  - $d\theta_1$, $d\theta_2$, $d\theta_3$, $d\theta_4$, $d\theta_5$. (These speeds will be used by Nextstate function)
- Wheels speed: *4 values*
  - u_1, u_2, u_3, u_4. (These speeds will be used by Nextstate function)
- Error Twist: *1 matrix*
  - X_{error} will be plotted in a graph.
- Added error twist this time:
  - X_{error} will accumulate and added in twist.

Approach:

1. Take input's chassis configuration to compute current X.
2. Find X_{error} and Adjoint by current X and X_{desire}. Find V_d by X_{desire} and $X_{desire,next}$.
3. Find twist by Adjoint, Kp, Ki, X_{err} and $X_{error,add}$.
4. Find Je.
5. Test if there is any joint will collide with each other by applying Je.
6. Find wheels and arms speed.

Test joint limit

Function

3 Input:

- Chassis configuration: *12 values*

  φ, x, y, θ_1, θ_2, θ_3, θ_4, θ_5, w_1, w_2, w_3, w_4
- Je: *1 matrix*

  obtained by FeedbackControl function.
- twist V:

  obtained by FeedbackControl function.

1 Output:

- Violate joint:

  this outputs which joint violate the joint limit

Approach:

1. Compute arms speed by Je and V.
2. Compute what is the joint angle after moving by this Je.
3. By using Scene 3, I assigned each joint upper and lower limit.
4. If joint's angle exceed limit, the column of Je will be zero.

Final Step: completing project

Full program

6 Input:

- Cube's initial position: *1 matrix*

  $T_{sc,initial}$ (transformation matrix from space to cube's initial position)
- Cube's final position: *1 matrix*

  $T_{sc,final}$ (transformation matrix from space to cube's final position)
- Actual initial configuration: *12 values*

  φ, x, y, θ_1, θ_2, θ_3, θ_4, θ_5, w_1, w_2, w_3, w_4
- Reference initial configuration: *12 values*

  there are some error between actual and reference initial configuration

1 Output:

- finalist: *13 values*

  φ, x, y, θ_1, θ_2, θ_3, θ_4, θ_5, w_1, w_2, w_3, w_4, (0 or 1)

Approach:

1. Find $T_{initial,reference}$ by reference initial configuration, and feed Trajectory generator function T, $T_{cube,initial}$ and $T_{cube,final}$ to generate trajectory matrices.
2. Compute $T_{initial,actual}$, and then find wheels and arms speed by feeding FeedbackControl function $T_{initial,actual}$ and X_{desire}, $X_{desire,next}$ (from trajectory matrices).
3. Feed Nextstate function to find next state. Save the next state data as finallist, and then feed next state back to FeedbackControl function again.
4. Assign gripper closing and opening time by output of Trajectorygenerator.
5. plot X_{error} and save finalist as csv file.

Results

Actual initial chassis configuration:

[-0.2 -0.2 0.1 -0.1 -0.2 -0.5 -0.1 0.1 0 0 0 0]

Reference initial chassis configuration:

[0 0 0 -0.3 -1 -0.5 -0.5 -0.5 0 0 0 0]

Initial cube position:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

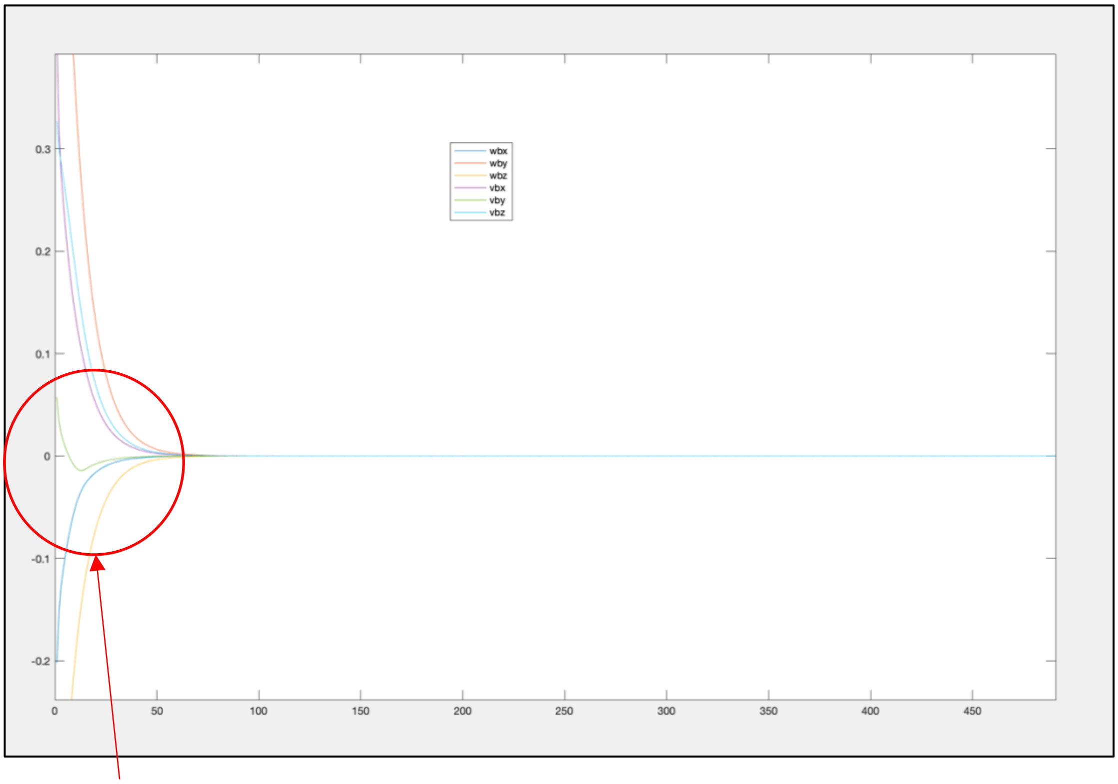
Final cube position:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Best

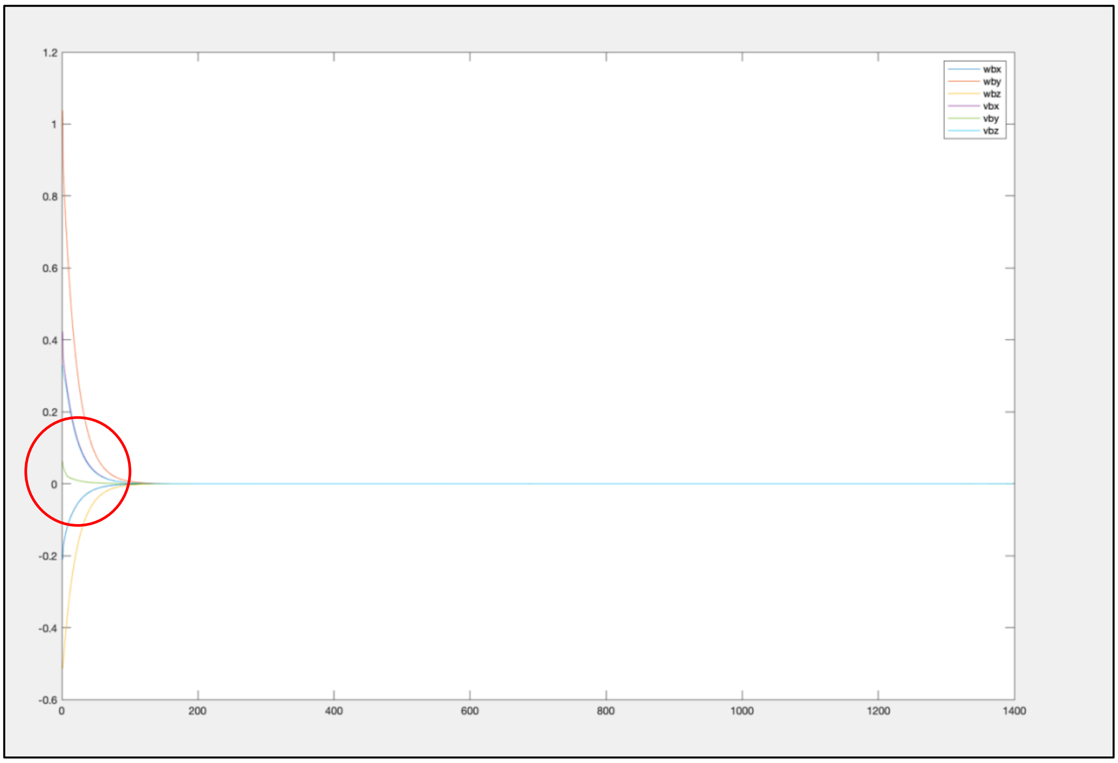
To obtain best performance, the following are trial and error of Kp:

- Kp=0, Ki=0

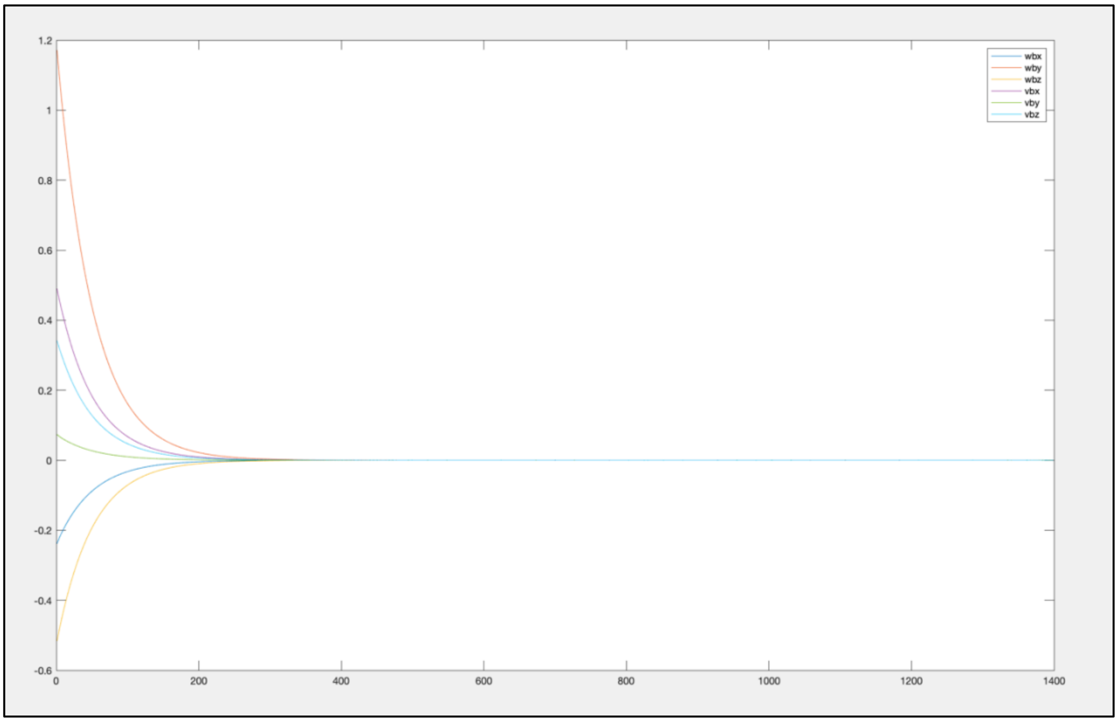


unsmooth curve

- Kp=0.5



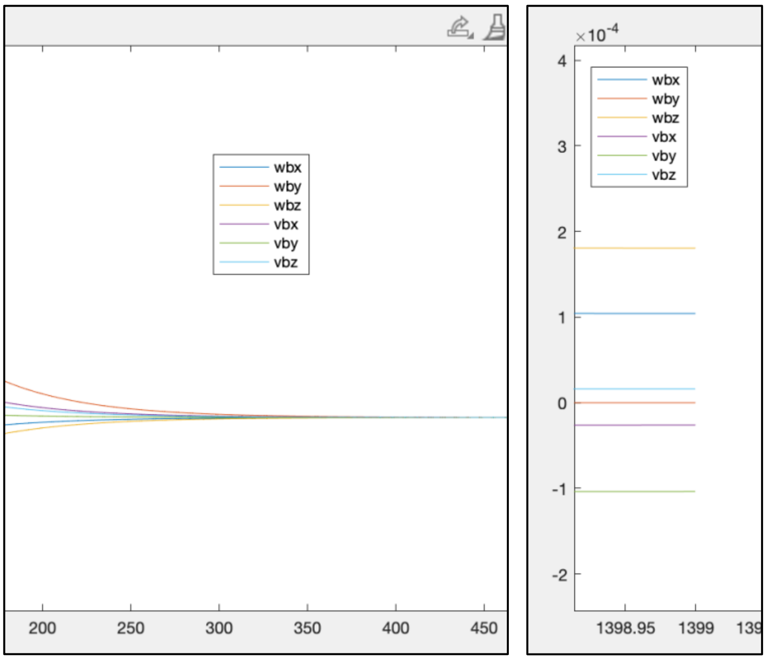
- Kp=0.2, Ki=0



I found let Kp=0.2, the transient responses are both smooth and clearly visible.

Besides, setting Ki=0 provides the smallest error twist at the end of time. The following is the comparison between applying the same Kp=0.2, but different Ki. Even very small Ki can affect the final error twist severely.

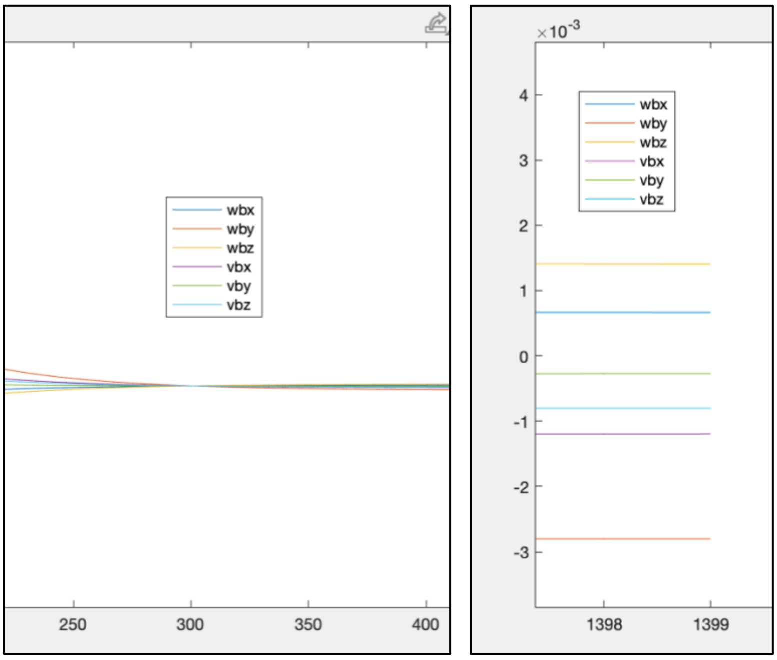
- Kp=0.2, Ki=0



Error twist < $2x10^{-4}$

Converge at almost a point

- Kp=0.2, Ki=0.0001



Error twist < $2x10^{-3}$

(an order larger)

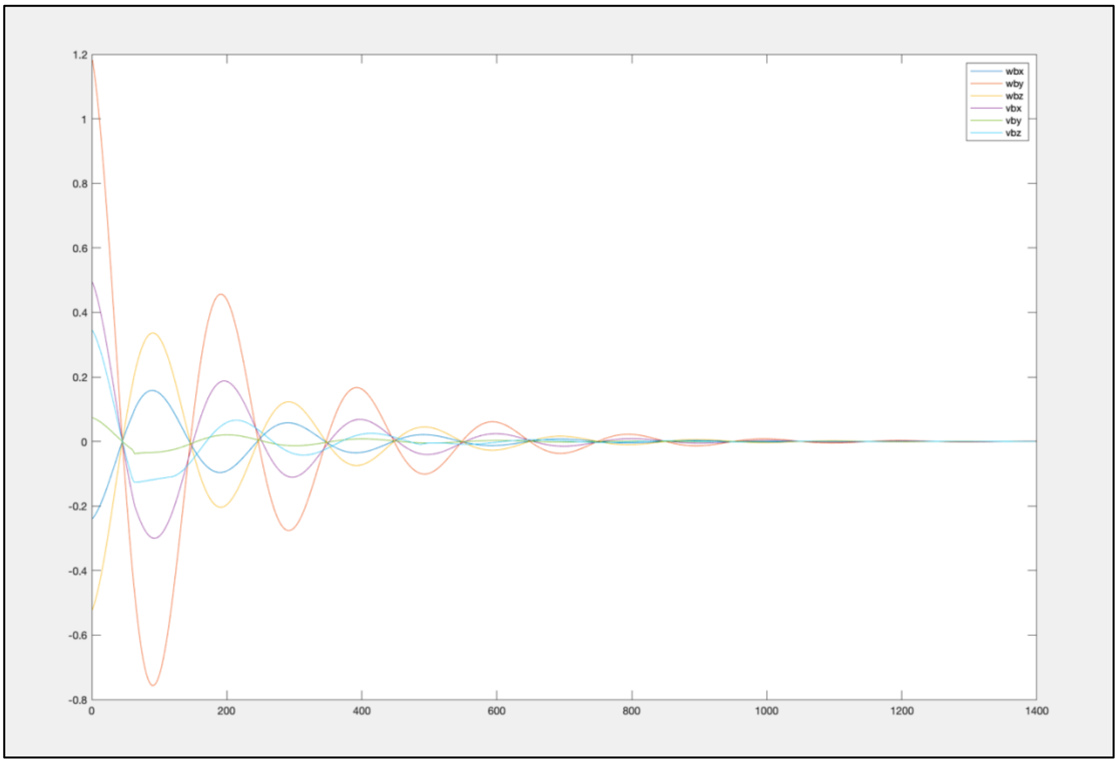
seems a little

bit overshoot

Therefore, my best controller is Kp=0.2, Ki=0.

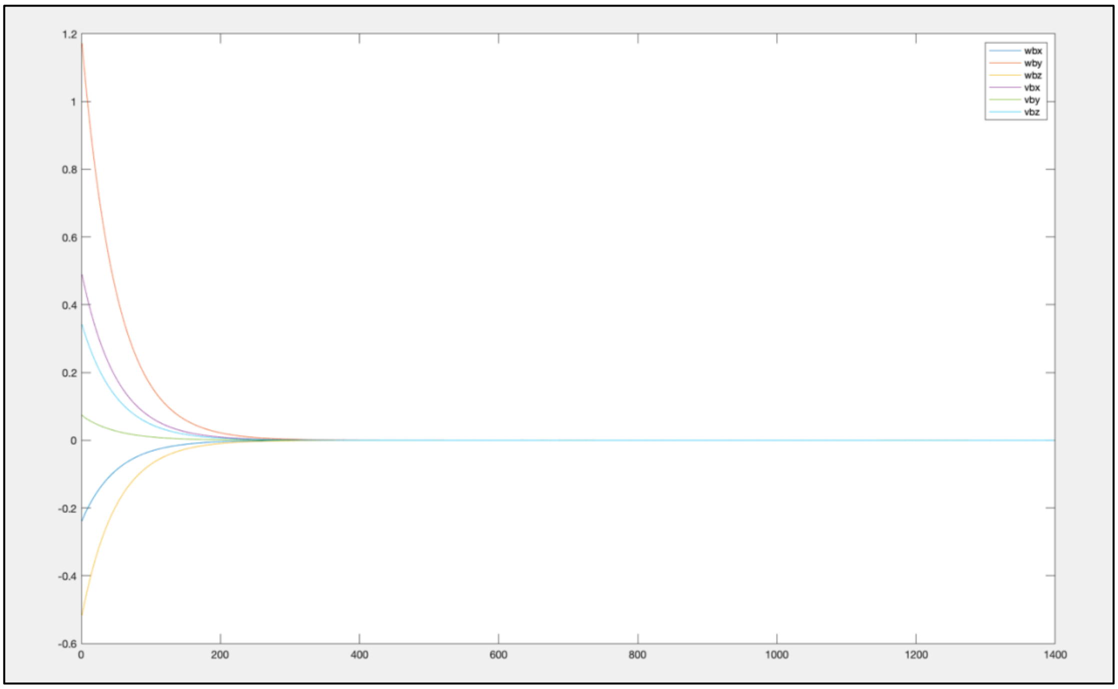
Overshoot

- Kp=0.1, Ki=0.1



New Task

- Kp=0.2, Ki=0



Video Link

Cube in trial position

"Best": Kp=0.2, Ki=0

Initial cube position:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Final cube position:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

link:https://drive.google.com/file/d/1Br5arklRjhyXk8Dib2lw40NR976s7xU3/view?usp=sharing

"Overshoot": Kp=0.1,Ki=0.1

link:https://drive.google.com/file/d/12hgEBS85sDvQ2HwolheuMGdPIldrDmMG/view?usp=sharing

Cube in new position

Initial cube position:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Final cube position:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

link: https://drive.google.com/file/d/1KW74QzLXuEg6D3NJO-FDx9RPU_9B2TNh/view?usp=sharing