EE683 Robot Control: Assignment4

1st Donghee Han
Robotics Program
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
hdh7485@kaist.ac.kr

***Index Terms*—Control, Robot, Disturbance Observer, Robust Internal-loop Compensator, DOB, RIC, Flexible Joint Robot, FJR, SEA**

I. Kinematically Redundant Robots

When the configuration of the robot is n-dimensional and the task variable that the robot performs is m-dimensional, it can be said to be kinematic redundant when $n > m$ is established. If robot become a kinematic redundant, it can do additional task. For example, robot can avoid singularity or avoid self-collision. If $n = m$ indicates that it can be executed and $n < m$ indicates that it cannot be executed.

$$\dot{p} = J(q)\dot{q} \tag{1}$$

For a simple example, if the robot has six joints and a work space is a six-dimensional vector consisting of three transitions and three orientations, Jacobian matrix is a 6×6 matrix. And if there is a reverse of Jacobian matrix, then inverse kinematics can also be calculated. When the workplace is called translation excluding rotation, the work space becomes three dimensions, and the Jacobian matrix becomes 3×6 dimensions.

As such, the $J(q) \in \mathbb{R}^{m \times n}(n > m)$ matrix is called has a non-trivial null-space. The jacobian matrix with non-trivial null-space can move each joint without motion on the end-effector. As such, the existence of a joint velocities without affecting the task velocities at the kinematic level is called null motion.

$$\dot{p} = J(q)\dot{q} = 0 \tag{2}$$

A. Moore-Penrose Pseudo-Inverse

A typical inverse matrix is $R^{n \times n}$, but non-trivial null-space Jacobian matrix is $R^{m \times n}$ instead of $R^{n \times n}$, so inverse matrix cannot be obtained in a typical way. More-Penrose pseudo-inverse (pseudo-inverse) should be used to obtain a inverse of $R^{m \times n}(n > m)$ (fat Matrix). Pseudo-inverse of matrix $A \in \mathbb{R}^{m \times n}(n > m)$ is expressed as A^+.

$$A^+ = A^T(AA^T)^{-1} \tag{3}$$

A^+A has various properties. A^+A can provides a lease-norm solution for a least square problem, and A^+Ax is a projection of a vector $x \in \mathbb{R}^n$ to row space of $A \in \mathbb{R}^{m \times n}$. A^+Ax is a component of column space, $(I - A^+A)x$ can be a vector orthogonal to A^+Ax.

B. Minimal parameterization of null-motion

If apply pseudo-inverse to non-trivial null-space Jacobian, it can be divide into two components: row space component (row projection matrix J^+J), null space component (orthogonal projection matrix $I - J^+J$).

When the Jacobian matrix is fat matrix, joint space can be decomposition into row-space and null-space follows:

$$\dot{q} = J^+(q)J(q)\dot{q} + (I - J^+(q)J(q))\dot{q} \tag{4}$$

To minimize the parameters of null-space, suppose that there is a matrix $Z(q) \in \mathbb{R}^(n - m) \times n$ that satisfy the following conditions:

$$\begin{aligned} J(q)Z(q)^T &= 0 \\ Z^+(q)Z(q) &= I - J^+(q)J(q) \end{aligned} \tag{5}$$

According to the (5), row space of $Z(q)$ spans null-space of $J(q)$. The result of parameter minimize follows:

$$\begin{aligned} &\begin{bmatrix} \dot{p} \\ \dot{n} \end{bmatrix} = \begin{bmatrix} J(q) \\ Z(q) \end{bmatrix} \dot{q} \\ &\leftrightharpoons \quad \dot{q} = \begin{bmatrix} J^+(q) & Z^+(q) \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{n} \end{bmatrix} \end{aligned} \tag{6}$$

For more generalization, add weight matrix $W(q)$ next to $Z(q)$. The $W(q)$ matrix is the $W(q) = W(q)^T > 0$ positive definite matrix.

$$\begin{aligned} J^{W+}(q) &= W^{-1}J^T\left(JW^{-1}J^T\right)^{-1} \\ Z^{W\#}(q) &= Z^T(q)\left\{Z(q)WZ^T(q)\right\}^{-1} \end{aligned} \tag{7}$$

By substituting it for weighted pseudo inverse the following results are obtained:

$$\dot{q} = \left[J^{W+}(q)Z^{\#}(q)\right] \begin{bmatrix} \dot{p} \\ \dot{n} \end{bmatrix} \tag{8}$$

$J^{W+}(q)$ is $J(q)$ right inverse and $Z^{W\#}$ is $Z(q)W(q)$ is right inverse. Use Jacobian matrix null space matrix Z and to decomposition the joint space again:

$$\dot{q} = J^{W+}(q)J(q)\dot{q} + \left(I - J^{W+}(q)J(q)\right)\dot{q} \quad (9)$$

One thing worry about is that augmentation of $Z(q)$, lost rank algorithmic singularity is that it can occur. But the result of determinant is not.

Convert the joint space to task space dynamics by substituting weighed pseudo inverse. $W(q) = M(q)$ for weight matrix, then $\Lambda_{pn} = 0$. And the inertial coupling between null dynamics and task dynamics is disappeared.

$$\begin{aligned} &\begin{bmatrix} \Lambda_p(q) & 0 \\ 0 & \Lambda_n(q) \end{bmatrix} \begin{bmatrix} \ddot{p} \\ \ddot{n} \end{bmatrix} \\ &+ \begin{bmatrix} \Gamma_p(q,\dot{q}) & \Gamma_{pn}(q,\dot{q}) \\ \Gamma_{np}(q,\dot{q}) & \Gamma_n(q,\dot{q}) \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{n} \end{bmatrix} \\ &+ \begin{bmatrix} \zeta_p(q) \\ \zeta_n(q) \end{bmatrix} \\ &= \begin{bmatrix} f \\ \eta \end{bmatrix} \end{aligned} \quad (10)$$

With weighted pseudo inverse, the point torque can be calculated as follows:

$$\begin{aligned} \tau &= \begin{bmatrix} J(q) \\ Z(q)W(q) \end{bmatrix}^T \begin{bmatrix} f \\ \eta \end{bmatrix} \\ &= J^T f + WZ^T \eta \end{aligned} \quad (11)$$

$$WZ^T\eta = \left[I - J^T(q)\left\{J^{W+}\right\}^T\right]\tau_{null} \quad (12)$$

τ_{null} means a joint torque that does not affect the task variable. For example, the manipulator can be controlled without affecting the acceleration of the task-dynamics.

II. Null Space Controller

We can suppression the null motion using inverse-dynamics control. Inverse dynamics controllers f and η is following:

$$\begin{aligned} f &= \Lambda_p u_p + \Gamma_p \dot{p} + \Gamma_{pn}\dot{n} + \zeta_p \\ \eta &= \Lambda_n u_n + \Gamma_{np}\dot{p} + \Gamma_n \dot{n} + \zeta_n \end{aligned} \quad (13)$$

And If substitute (13) for (10), we can get the following results.

$$\begin{pmatrix} \ddot{p} \\ \ddot{n} \end{pmatrix} = \begin{pmatrix} u_p \\ u_n \end{pmatrix} \quad (14)$$

If $\dot{n}_{des}$ is 0, control input u_n is following:

$$u_n = -K_{n,d}\dot{n} - K_{n,p}\int \dot{n} \quad (15)$$

and the result,

$$\begin{aligned} \ddot{p} &= u_p \\ \dot{n} &\to 0 \end{aligned} \quad (16)$$

To project joint damping to the null space, we can utilize τ_{null} in the control input (11). τ_{null} does not influence $\ddot{p}$ because of projector matrix (12). To apply the joint damping without affecting the task space, we can substitute $D\dot{q}$ for τ_{null}.

When ordering the robot to take a second task using null motion for $V(q)$, you can substitute $\kappa\nabla V(q)$ for τ_{null}. For example, self-collision, Singularity, distance from an object, etc. With this, null-motion attempts to maximize $V(q)$.

III. Hierarchical control approach

When a robot has to do several tasks at the same time, each task may collide with each other and not be able to work at the same time. In this case, the priority of the task can be determined and ordered to perform from the highest priority.

For example, when humanoids do a number of different tasks, the highest priority is to center of gravity. If the center of gravity is not held, it can fall down without doing other tasks.

Augmented Jacobian matrix is required to perform several tasks. Augmented Jacobian matrix is a matrix consisting of Jacobian matrix for each task.

$$J_j^{aug} = \begin{pmatrix} J_1 \\ \vdots \\ J_j \end{pmatrix} \quad (17)$$

As a result, control input τ is sum of each task's torque and gravity.

$$\begin{aligned} \tau &= g + \sum_{i=1}^{r} \tau_i \\ \tau_i &= N_i J_i^T F_i \end{aligned} \quad (18)$$

If parameterize the null motion each task, the velocity of each task v is the product of null-motion Jacobian matrix $\bar{J}$ and joint velocity $\dot{q}$.

$$v = \bar{J}\dot{q} \quad (19)$$

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + \tau_{ext} \quad (20)$$

And substitute $v = \bar{J}\dot{q}$ for robot dynamics (20) is following:

$$\Lambda\dot{v} + \Gamma v = \begin{pmatrix} F_1 \\ [\text{ projection }_2]\, J_2^T F_2 \\ \vdots \\ [\text{ projection }_r]\, J_r^T F_r \end{pmatrix} + \bar{J}^{-T}\tau_{ext} \tag{21}$$

In the (21), the inertia matrix Λ is a block diagonal matrix and $\bar{J}^{-T}$ is upper triangular matrix.

If external interaction is applied, external wrench allocated with ith task, it influences priorities that higher than i. In other words, it is the opposite of control input.

$$\bar{J}^{-T}\tau_{\text{ext}} = \bar{J}^{-T}\left(J_r^{\text{aug}}\right)^T F_{\text{ext}} \tag{22}$$

IV. Under Actuated Robot

The formal definition of under actuated robot depends on input-affine nonlinear system.

$$\dot{x} = f(x) + g(x)u \tag{23}$$

If following condition is be established, it is under actuated system.

$$rank[g(x)] < dim[q] \tag{24}$$

Typical examples of under actuated systems are humanoids, drones, and carts.

A. Flexible Joint Robot

The joints of robots have elasticity because of mechanic power transfer or elasticity of material. To calculate additional elasticity, a dynamics with elasticity must be modeled. Otherwise, problems such as vibration and poor performance can occur.

The flexible joint robot(FJR) is an underactuated mechanical system with elasticity. FJR has link side dynamics and actuator side dynamics, and link side dynamics has to be controlled by actuator side commands. In other words, non-collocated command inputs and dynamic effects to be controlled.

The system dynamics of the basic FJR model are as follows.

$$\begin{aligned} &B\ddot{\theta} + K_j(\theta - q) = \tau_m \\ &M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = K_j(\theta - q) + \tau_{ext} \end{aligned} \tag{25}$$

To check the FJR model is under-actuated, we should check the rank of $g(x)$ and dimension of state x.

$$x = \begin{pmatrix} \theta \\ \dot{\theta} \\ q \\ \dot{q} \end{pmatrix} \tag{26}$$

The rank of $g(x)$ is n and dimension of state is $2n$. Therefore, the FJR model is an under actuated system.

In the FJR model, inertial coupling exists between motor and link with some assumptions. There is small displacement at joints because of hook's law, and axis-balanced motors. Each motor is mounted on the robot in a position preceding the driven link and angular kinetic energy of motor is due only when it spin. This assumptions can be often neglected in practice, especially when large gear reduction is used. These assumptions can actually be ignored. Especially when using a large gear reducer.

An important variable in the FJR model (31) is the joint torque $K_j(\theta - q)$. Because if we can not measure the joint torque, we can not utilize the FJR model. In practice, there are two methods two measure the joint torque. For low stiffness case, displacement of joint can be measured using two encoders, For high stiffness case, joint torque can be measured using joint torque sensor.

To PD control the FJR stable, we should know link position q and motor velocity $\dot{theta}$ or motor position θ and motor velocity $\dot{\theta}$ for feedback. If link position and motor velocity are used, it is asymptotically stable for $0 < k_p < k_j, k_d > 0$. And if motor position and motor velocity are used, it is asymptotically stable for $k_p > 0, k_d > 0$.

B. Motor Position PD Controller

The goal of motor position PD controller is regulating link position q to desired position q_d. The FJR model dynamics is same as (31), and control is following:

$$\begin{aligned} &\tau_m = K_P(\theta_d - \theta) - K_D\dot{\theta} + g(q_d) \\ &\theta_d = q_d + K_j^{-1} g(q_d) \end{aligned} \tag{27}$$

According to [2], if following condition is established, the closed-loop equilibrium point $(\theta_d, 0, q_d, 0)$ is globally asymptotically stable.

$$\lambda_{\min}(K_E) = \lambda_{\min}\left(\begin{bmatrix} K_j + K_P & -K_j \\ -K_j & K_j \end{bmatrix}\right) > \alpha \tag{28}$$

But there are some limitation of motor-feedback PD controller. PD controller is highly affected by friction. Moreover, if human interact with the pd controlled robot, human can feel the motor-inertia by gear reduction.

$$\begin{aligned} &B\ddot{\theta} + K_j(\theta - q) = \tau_m + \tau_f \\ &M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = K_j(\theta - q) + \tau_{ext} \end{aligned} \tag{29}$$

In the (29), replace the motor torque τ_m as follow:

$$\tau_m = \tau_j + BB_\theta^{-1}(-\tau_j + u) \tag{30}$$

Then (29) is organized as follows:

$$\begin{aligned} &B\ddot{\theta} + K_j(\theta - q) = \tau_m \\ &M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = K_j(\theta - q) + \tau_{ext} \end{aligned} \tag{31}$$

It is same system with (29) with new control input u. The smaller the B_θ, the smaller the disturbance such as friction.

V. Two time scale analysis

Suppose that joint of FJR is very stiffness. The joint stiffness K_j in (31) will be very large. To change the model to two-time scale model, replace the state (θ, q) to $(q, z = K_j(\theta - q))$. State z is a fast variable of two-time scale model. It is a convention to use z in two-time scale modeling. The expression of FJR dynamics with the changed state is as follows:

$$\begin{aligned} \ddot{q} &= M^{-1}(-C\dot{q} - g + z) \\ \ddot{z} &= K_j B^{-1}\tau_m + K_j M^{-1}(C\dot{q} + g) \\ &\quad - K_j\left(B^{-1} + M^{-1}\right) z \end{aligned} \tag{32}$$

The J_j is so large that it can be replaced as $\frac{1}{\epsilon^2}I(\epsilon \ll 1)$ for simplification.

$$K_j = \frac{1}{\epsilon^2} I \tag{33}$$

Using (33) If use (33) to express the (32), it will be as follows:

$$\begin{aligned} \ddot{q} &= M^{-1}(-C\dot{q} - g + z) \\ \epsilon^2\ddot{z} &= B^{-1}\tau_j + M^{-1}(C\dot{q} + g) - \left(B^{-1} + M^{-1}\right) z \end{aligned} \tag{34}$$

Two-time scale dynamics need a new time scale σ to create fast and slow dynamics.

$$\begin{aligned} \sigma &= \frac{t}{\epsilon} \\ \epsilon\frac{d}{dt}z &= \frac{d}{d\sigma}z \end{aligned} \tag{35}$$

New time scale σ is faster $\frac{1}{\epsilon}$ times than original time scale t.

New time scale σ, not t, expressed as ' the differential variable, we can describe the slow dynamics and fast dynamics as following:

$$\begin{aligned} \text{Slow dynamics} : q'' &= \epsilon^2\left(M^{-1}(-C\dot{q} - g + z)\right) \\ \text{Fast dynamics} : z'' &= B^{-1}\tau_m\big|_{\epsilon=0} + M^{-1}(C\dot{q} + g) \\ &\quad - \left(B^{-1} + M^{-1}\right) z \end{aligned} \tag{36}$$

And when ϵ is so small that it says zero, the slow dynamic of (36) becomes $q\prime\prime = 0$. That is $q, \dot{q}$ is frozen variable in fast time scale. Fast dynamics, on the other hand, has z'' without erasing.

Original system can be divided into boundary layer system and reduced system when ϵ =0. Fast dynamics (36) is called the boundary-layer system.

$$\ddot{q} = M^{-1}(-C\dot{q} - g + h) \tag{37}$$

When $z = h(t,x)$ of fast dynamics is equilibrium, $z = h(t,x)$ of slow dynamics from $\epsilon = 0$ is referred to as the reduced system (37).

A. Control Design based on Two-time Scale approach

FJR dynamics (31) is expressed in fast and slow dynamics:

$$\begin{aligned} \ddot{q} &= M^{-1}(-C\dot{q} - g + z) \\ \epsilon^2\ddot{z} &= B^{-1}\tau_m + M^{-1}(C\dot{q} + g) - \left(B^{-1} + M^{-1}\right) z \end{aligned} \tag{38}$$

Motor torque, which is control input, is divided into slow control and fast control and defined as follows.

$$\tau_m = \tau_s(q, \dot{q}, t) + \epsilon\tau_f(q, \dot{q}, z, \dot{z}, t) \tag{39}$$

Slow control τ_s is the same reduced model as rigid dynamics in slow time scale t. Fast control τ_f is designed to control movement that occurs with elasticity in fast dynamics. When giving feedback on two models divided by time scale, it is not necessary to combine them into one.

$$\begin{aligned} &\epsilon^2\ddot{z} + B^{-1}K_f\dot{z} + \left(B^{-1} + M^{-1}\right) z \\ &= B^{-1}\tau_s + M^{-1}(C\dot{q} + g) \end{aligned} \tag{40}$$

In the boundary layer system, (40) is exponentially stable and z is:

$$z = (B^{-1} + M^{-1})^{-1}(B^{-1}\tau_s + M^{-1}(C\dot{q} + g)) \tag{41}$$

And if you substitute (41) for z of slow dynamics:

$$(M(q) + B)\ddot{q} + c(q, \dot{q})\dot{q} + g(q) = \tau_s(q, \dot{q}, t) \tag{42}$$

Now we must design the reduced model controller with slow dynamics (42). The reduced model is the original rigid-point robot dynamic.

$$\begin{aligned} \tau_s &= (M(q) + B)(\ddot{q}_d + K_D(\dot{q}_d - \dot{q}) \\ &\quad + K_P(q_d - q)) + C(q, \dot{q})\dot{q} + g(q) \end{aligned} \tag{43}$$

The following results are obtained by substituting slow dynamic (43) for (42).

$$\ddot{\tilde{q}} + K_D\dot{\tilde{q}} + K_P\tilde{q} = 0 \tag{44}$$

And (44) is exponentially stable.

B. Cascade Control

Cascade Control is an approach that controls the controller divided into several controllers. Consider the stability between the first controller$\dot{x_2} = f_2(x_2)$ and the second controller $\dot{x_1} = f_1(x_1)$.

1) Autonomous System:

$$\begin{aligned} \Sigma_1 &: \dot{x}_1 = f_1\left(x_1, x_2\right) \\ \Sigma_2 &: x_2 = f_2\left(x_2\right) \end{aligned} \tag{45}$$

If $x_2 = 0$ is asymptotically stable when $\dot{x_2} = f_2(x_2)$, and If $x_1 = 0$ is asymptotically stable when $(x_1, x_2) = 0$, then if $(x_1, x_2) = 0$, the original system is asymptotically stable. If $x_2 = 0$ is globally asymptotic stable when $\dot{x_2} = f_2(x_2)$, and If $x_1 = 0$ is globally asymptotically stable when $(x_1, x_2) = 0$, then if $(x_1, x_2) = 0$, the original system is globally asymptotically stable [3].

2) Non-autonomous System:

$$\begin{aligned} \Sigma_1 &: \dot{x}_1 = f_1\left(t, x_1\right) + g\left(t, x_1, x_2\right) x_2 \\ \Sigma_2 &: \dot{x}_2 = f_2\left(t, x_2, u\right) \end{aligned} \tag{46}$$

In a non-autonomous system, if Σ_2 is Global Uniform Asymptotic Satble and Σ_1 is Global Explicit Stable when x_2 =0.00, the original system is GUAS. If Σ_2 is GUAS and Σ_1 is GUAS when $x_2 = 0$, the original system is GUAS.

[1]

3) Inner-loop Joint Torque Controller: We can utilize the cascade control for joint torque control; inner-loop joint torque control. The goal of the inner-loop joint torque control is to change τ_j to $\tau_d(t, q, \dot{q}$. It can make to control the link side dynamics directly. First change the state (θ, q) to (q, τ_j), make an equation about joint torque $\ddot{tau}_j$:

$$\begin{aligned} \ddot{tau}_j = K_j B^{-1} \tau_m + K_j M^{-1}(C\dot{q} + g) \\ - K_j(B^{-1} + M^{-1})\tau_j \end{aligned} \tag{47}$$

To make the right term of the equation u, make τ_m as follows:

$$\begin{aligned} \tau_m = BK_j^{-1}u - BM^{-1}(C\dot{q} + g) + B(B^{-1} + M^{-1})\tau_j \\ u = \ddot{\tau}_d + L_D(\dot{\tau}_d - \dot{\tau}_j) + L_p(\tau_d - \tau_j) \end{aligned} \tag{48}$$

Then we can get an external stable controller as follows.

$$\left(\ddot{\tau}_j - \ddot{\tau}_d\right) + K_D\left(\dot{\tau}_j - \dot{\tau}_d\right) + K_P\left(\tau_j - \tau_d\right) = 0 \tag{49}$$

And it becomes link-side dynamics as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau_d \tag{50}$$

If (50) is GUAS or AS, the original system is also GUAS or AS system. However, to calculate $\ddot{\tau}_d$ for (48), there is a limit to know $\dddot{q}$.

VI. Quadrotor Control

A. Quadrotor dynamics

The translation dynamics of the quadrotor are expressed as mge_3 in gravity and fRe_3 in rotor thrust.

$$\begin{aligned} \dot{p} &= v \\ m\dot{v} &= mge_3 - fRe_3 \end{aligned} \tag{51}$$

And the rotation dynamics of the quadrotor are as follows.

$$\begin{aligned} \dot{R} &= R\hat{\Omega} \\ J\dot{\Omega} + \Omega \times J\Omega &= M \end{aligned} \tag{52}$$

The thrust $[f_1, f_2, f_3, f_4]$ of each motor multiplied by the allocation matrix results in the sum of thrust, roll, pitch, yaw $[f, M_1, M_2, M_3]$.

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{53}$$

Torque $\tau_i = (-1)^i c_{\tau f} f_i$ is generated when the propeller rotates. If the torque is not removed, the quadrotor will continue to rotate, thus counterclockwise for the third and fourth to offset the torque.

In the translation motion $m\dot{v} = mge_3 - fRe_3$ of quadrotor, f is the control input and the down direction of quadrotor Re_3 is associated with the position of the quadrotor. In other words, the rotation of the quadrotor must be changed in order to move the translation. Thus, a typical drone can be seen as an under-actuated system. Full-actuated can be implemented if more than six propellers (the more, the better).

B. Differential Flatness

Flat system is a system that can express the output and derivatives of a system as state and input. When you have the following systems:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x) \end{aligned} \tag{54}$$

If $z, z^2, ..., z^{(k)}$ is exist 1-1 mapping with $x(t), u(t)$, then z is called flat output. The quadrotor has $k = 4$. When the flat system is used in the quadrotor, all states and inputs can be defined with four time derivatives of output $\sigma(t)$. That is, the position and rotation direction can be defined with four differentials, and the trajectory can be sufficiently created in four dimensions instead of 12 dimensions.

C. Geometric quadrotor control

The goal of the geometric quadrotor control is to fly along a given trajectory σ.

$$\sigma(t) = (x(t), y(t), z(t), \psi(t)) \tag{55}$$

The quadrotor must obtain thrust f and the quadrotor's orientation $\vec{b}_{3d}(t)$ in order to follow the trajectory. $\vec{b}_{1d}t)$ is desired yaw, so we can define rotation matrix. Track the orientation of the trajectory with the rotation matrix using thrust f and track the position.

To calculate the rotation matrix, we should know a unit vector of $\vec{b}_{3d}t$ and desired yaw vector $\vec{b}_{1d}$.

$$\begin{aligned} f_{des} &= -k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d \\ e_x &= p - p_d \\ e_v &= v - v_d \end{aligned} \tag{56}$$

Cross product of $\vec{b}_{3d}t$ and $\vec{b}$ follows $\vec{b}_{2d}(t)$ and calculate Proj$[\vec{b}_{1d}t)]$ to obtain desired rotation matrix R_d.

$$\vec{b}_{3d} = -\frac{-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d}{\|-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d\|} \tag{57}$$

Then use f_{des} to control the position.

When SO(3) PD control is added, the feed-forward term of speed and acceleration is as follows.

$$\begin{aligned} &J\dot{\Omega} + \Omega \times J\Omega = M \\ &M = \Omega \times J\Omega - k_R \log(\tilde{R}) \\ &- k_\Omega e_\Omega - J\left(\widehat{\Omega} R^T R_d \Omega_d - R^T R_d \dot{\Omega}_d\right) \\ &J\dot{e}_\Omega + k_\Omega e_\Omega + k_R \log(\tilde{R}) = 0 \end{aligned} \tag{58}$$

(58) is exponentially stable.

In practice, M of (58) can be implemented very simply. The simplified M is as follows.

$$M = -k_R e_R - k_\Omega e_\Omega \tag{59}$$

$\Omega \times J\Omega - k_R$ in (58) can be removed from the model, and $\log(\tilde{R})$ can be calculated as follows:

$$\log(R_d^{-1} R) = \frac{\theta}{2\sin(\theta)}(R_d^T - R^T R_d) \tag{60}$$

However, $\frac{\theta}{\sin(\theta)}$ on (60) can be k_R and e_R will eventually be:

$$e_R = \frac{1}{2}\left(R_d^T R - R^T R_d\right)^\vee \tag{61}$$

And even if you remove the feed forward term, it works well if you tune the PD gain well, so remove it for simplicity. This results as shown in (59).

VII. Experiment

In this experiment, null space controller was implemented with 2DOF manipulator. Added damping to the joint without affecting the task dynamics by using a null space.

The two links of the manipulator are set at 1.5 and the target x position x_d is set at 1.5. And the simulation time is from 0 to 60 seconds and the control cycle is 0.001 seconds, or 1 kHz.

At first, manipulator was used only task space without null space and controlled task dynamics with x position. With this control, the end-effector's x position converges at x_d, but the angle of each point and the end-effector's y-position continue to change.

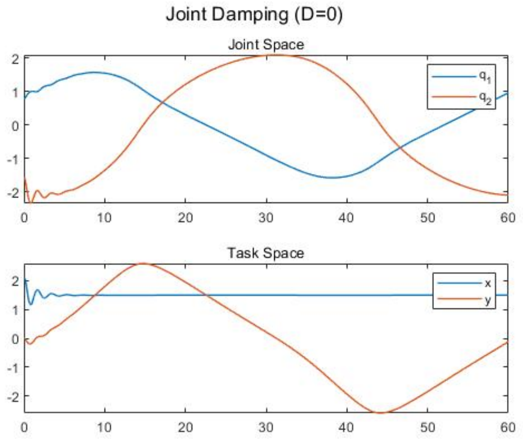

Fig. 1. The angle of joints and position of end-effector when the damping coefficient is 0

As a result of adding point stamping (D=1.0) to the null space, it naturally converge to x_d and collects angle q of each joints from about 30 seconds. At this time, x of the task space continued to converge at 1.5 and the null space controller did not affect the task space at all.

In the same simulator environment, I compared the convergence position and the travel width by different damping factors. The damping coefficients tested are 0, 0, 0, 0, 1.0 and 1.5, respectively. First, when you look at the x position of Fig. 3 you can see that they all converge correctly at x_d. The graph is almost the same, indicating that the null space controller did not affect the task space at all. On the other hand, given the y position, the greater the damping factor, the less the y movement. In other words, it could be seen that the null space controller only controlled the null space without affecting the task space at all.

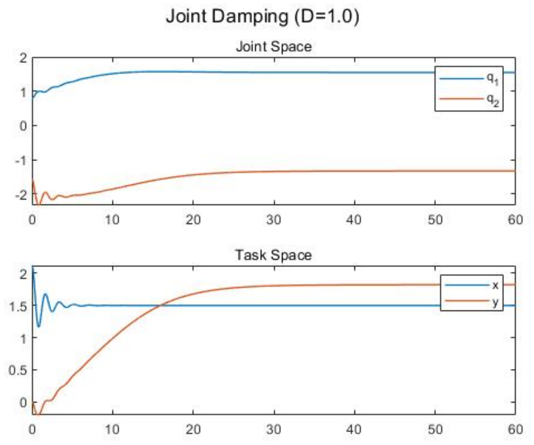

Fig. 2. The angle of joints and position of end-effector when the damping coefficient is 1.0

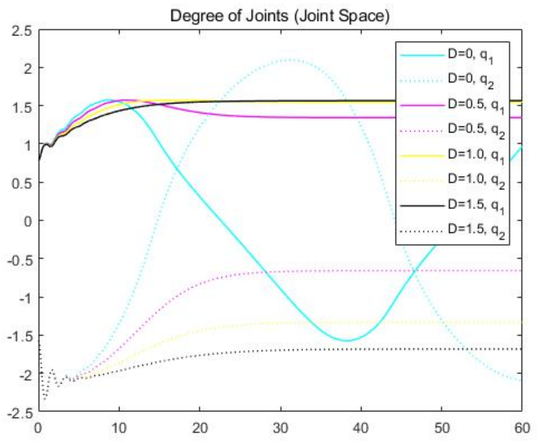

Fig. 4. Angle of joints q_1, q_2 about each damping coefficients. Solid lines are first joint and dotted lines are second joint.

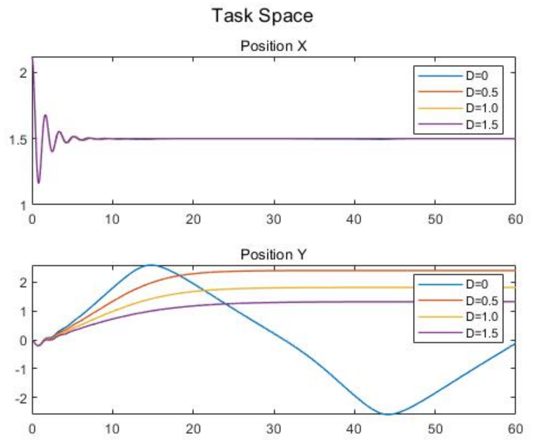

Fig. 3. The end effector position about each damping coefficient. Position X looks like one graph but actually, the 4 graphs were overlapped. In other words, null space controller didn't affect.

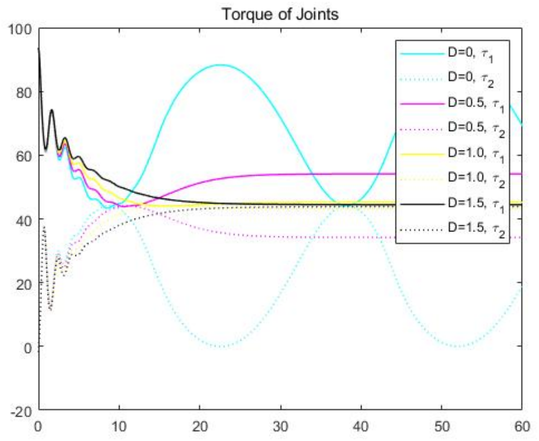

Fig. 5. The greater the damping factor, the slower the torque was slowly converging.

The greater the damping factor, the better the damping effect, given the change in the position of each joint. When the damping factor is zero, i.e. when the null space controller is not inserted, you can see that the q_1, q_2 joints continue to move around drawing wave forms. However, the higher the damping factor, the less movement of each joint. In other words, we could see that the point stamping was successful.

In Fig. 5 the larger the damping factor, the faster the convergence could be seen. When the coefficient is 0, it does not converge, and as the coefficient increases, the converging torque of the two joints becomes closer.

As the damping factor increases, it seems to converge well, so I tried to increase the damping factor to an extreme extent. When damping factor is $D = 10$ Fig. 6 the torque of the two motors appeared completely different. There is a disadvantage that control performance is very poor when one disturbance is applied.

References

[1] Elena Panteley and Antonio Loria. Global uniform asymptotic stability of cascaded non-autonomous nonlinear systems. In *1997 European Control Conference (ECC)*, pages 973–978. IEEE, 1997.

[2] Patrizio Tomei. A simple pd controller for robots with elastic joints. *IEEE Transactions on automatic control*, 36(10):1208–1213, 1991.

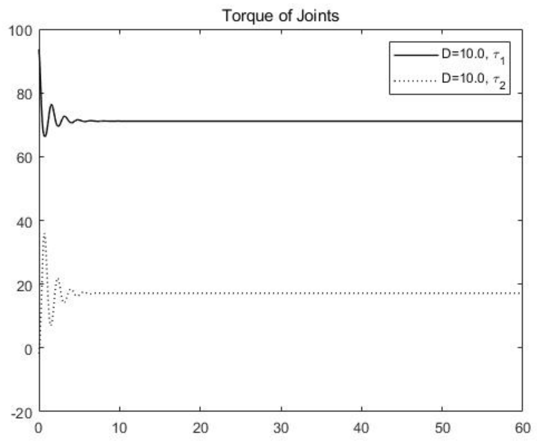

Fig. 6. The difference in motor torque was very large when the damping factor was extremely large.

[3] Mathukumalli Vidyasagar. Decomposition techniques for large-scale systems with nonadditive interactions: Stability and stabilizability. *IEEE transactions on automatic control*, 25(4):773–779, 1980.