Robot Mechanics Final Project

2012-11659 Hotae Lee, *Student, SNU M.E.*

I. ABB IRB 1200 MODEL IDENTIFICATION

THIS ABB IRB 1200-5/0.9 robot manipulator is set by the given specifications. Because it has only total mass, we assigned mass of each links by considering their sizes. We use a MATLAB Robotic toolbox from Perter Corke to simulate the graphic model of IRB 1200. Some torque controllers are simulated in Simulink S-function and displayed in the Robotic Toolbox. The length of each link is shown in Fig.1 and the mass and inertia moment we assigned is shown in Table1. We expressed every numerical data in SI Unit. To simplify the model, each link is assumed as cylinder while we calculated the inertia moment. In the Robotic Toolbox, it is created by SerialLink based on DH-parameter, which is shown as Table2. We denoted each joint angle as θ_i (i=1…6) and q=[θ_1;…; θ_6]. This m.file is named as *mdl_ABBIRB1200*.

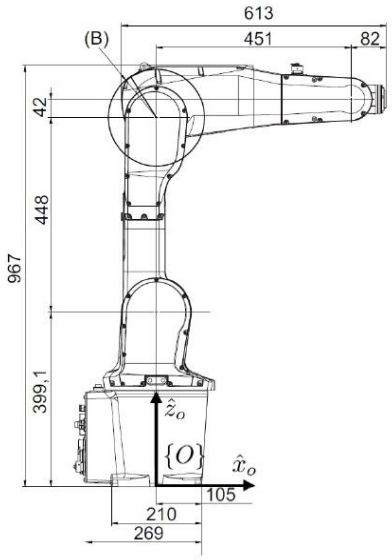

Fig 1. IRB 1200 manipulator's blueprint.

We placed the robot's based in the origin and the workpiece(h:120mm, D:60mm) is located in [0.45;0;0.35] and the laser is located in [0.400;0.400;0.600]. Also, the sanding belt is located in [0.650;0.200;0.450]. Based on this specification and DH-parameter, we set the initial pose of IRB 1200 robot manipulator as Fig.2. We denote a green cylinder as a workpiece and yellow box as a desk. While we accomplish the task, the laser and the sanding belt also will be described in the graphic display. We accomplished three tasks: 1) Picking up the workpiece 2) Removing rust by laser 3) Polishing with sanding belt.

TABLE 1
IRB 1200 SPECIFICATION (SI UNIT)

Link	Length(m)	Mass(kg)	Inertia Moment(kg*m^2)		
1	0.399	16	0.2122	0	0
			0	0.2122	0
			0	0	0.1764
2	0.448	15	0.2508	0	0
			0	0.0750	0
			0	0	0.1254
3	0.042	4	0.0006	0	0
			0	0.0050	0
			0	0	0.0006
4	0.451	16	No consider		
5	0.082	3	No consider		
6	0	0	No consider		

TABLE 2
DH-PARAMETER

Link	α	a	d	θ
1	π/2	0	0.399	Θ1
2	0	0.448	0	Θ2+ π/2
3	π/2	0.042	0	Θ3
4	- π/2	0	0.451	Θ4
5	π/2	0	0	Θ5
6	0	0	0.083	Θ6

The position and orientation of End-Effector (EF) can be described as the multiplication of transformation matrix $g_{l_{i-1}l_i}$, which is 4×4 matrix in SE(3) group. Each transformation matrix is calculated by DH-parameter method as below.

$$g_{st} = g_{sl_1}(\theta_1) \cdots g_{l_5 t}(\theta_6) \quad (1)$$

$$g_{l_{i-1}l_i} = Rot_{z_{i-1}}(\theta_i) Trans_{z_{i-1}}(d_i) Trans_{x_i}(a_i) Rot_{x_i}(\alpha_i) \quad (2)$$

Based on Table1 and Table2, we can calculate $g_{l_{i-1}l_i}$. Moreover, these multiplication of 4×4 matrices can be calculated by the *fkine* function in Robotic Toolbox. In Fig.3, we can confirm the initial pose of the 6-DOF manipulator with zero joint angles. Calculated transformation matrix of EF in initial position is shown as next..

$$g_{EF_{ini}} = \begin{bmatrix} 0 & 0 & 1 & 0.533 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.889 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

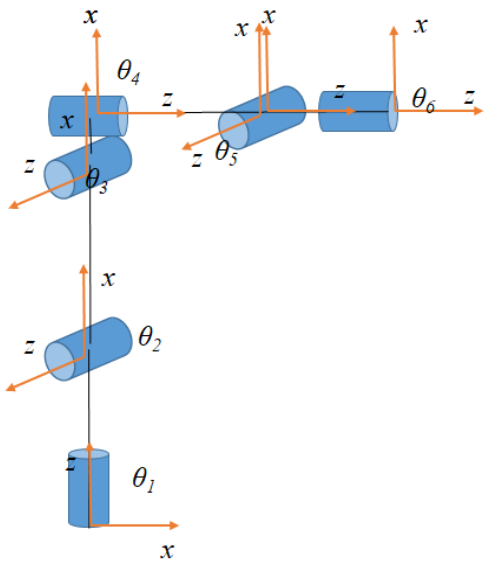

Fig 2. Simplified link model of IRB 1200

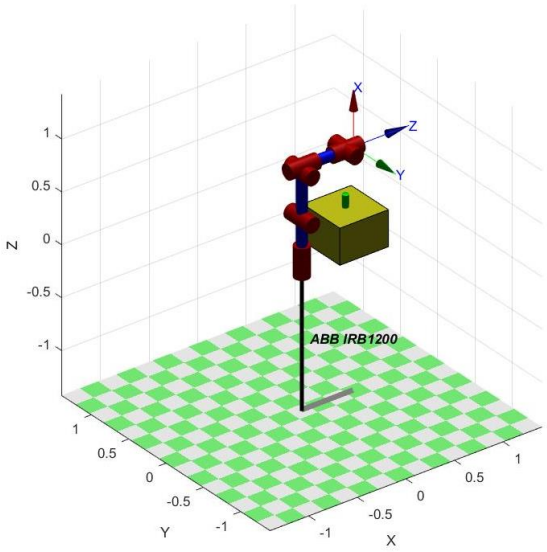

Fig 3. A created model in Robotic Toolbox (yellow : desk, green: workpeice)

We can find that the orientation of EF in Fig.3. is identical to transformation matrix's orientation in (3).

II. PICKING UP THE WORKPIECE

In order to pick up the workpiece, EF should be located on the upper side workpiece while both plane is coinciding. Thus, we found the inverse kinematics of the EF when the EF's position is [0.450;0;0.470] and the EF's z-axis direction is same as z-direction of fixed frame. x-axis direction and y-axis direction are not constrained. For convenience, we chose the Rotx(pi/2).

A. *The required Pose of the end-effector*

We appointed the desired transformation matrix in SE(3) and we could obtain the required joint angles through solving inverse kinematics. 6-DOF inverse kinematics problem is very difficult to solve it. Thus, we used *ikine* function to solve the inverse kinematics and the results are described as below. $g_{EF_{desired}}$ is the desired transformation matrix and q_{inv} is the solution of inverse kinematics. This desired pose in the model can be shown as Fig.4.

$$g_{EF_{desired}} = \begin{bmatrix} 1 & 0 & 0 & 0.45 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.47 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$q_{inv} = [0; -0.2191; -0.5523; 0; -0.7994; 0] \quad (5)$$

B. *Joint angle controller*

First, we considered the joint angle controller, which means that we can offer the joint angle input directly to the manipulator. Then, we should find the trajectory to get to the required configuration. In general, we are able to construct the arbitrary trajectory with boundary conditions with two given transformation matrix. However, in this Robotic Toolbox, there was an useful function. Since we already knew the previous SE3 and the required SE3, we could create the trajectory by using *ctraj* function. After we found the trajectory, we can simulate the manipulator with desired angle trajectory inputs. It can be shown in the video attached by email and simply described in Fig.5. Although it seems that the EF overlaps the workpiece, the controlled position is correct. This is because Joint size is set too large. We controlled the EF position, which is located in the middle of the last joint. So, it does not matter that it looks like overlapping. You can also check it in the submitted codes.

C. *Joint rate controller*

Next, we considered the joint rate controller. Since we cannot offer joint angle input directly, we should offer the appropriate angular velocity input to make the joint angles converge on desired angles. We designed the controller as below. (k=60)

$$\dot{q} = v \quad (6)$$

$$v = -k(q - q_d) \quad (7)$$

When we use this controller, q converges on q_d(desired angle) exponentially. So, we set the q_d same as the inverse kinematics following the trajectory we obtained above. We can obtain the present angle from the encoder with ± 0.05 resolution degree. This resolution degree is implemented through user-defined function. This function can be explained simply. When we get some numerical data, we multiply 100 and get the gauss function of that value. We round off that integer and divide it by 100. Through this process, all data can be recognized as ± 0.05 resolution. We named this function *resol* Function. Then, we updated the angle rate in 1kHz and simulated it in 1 seconds.

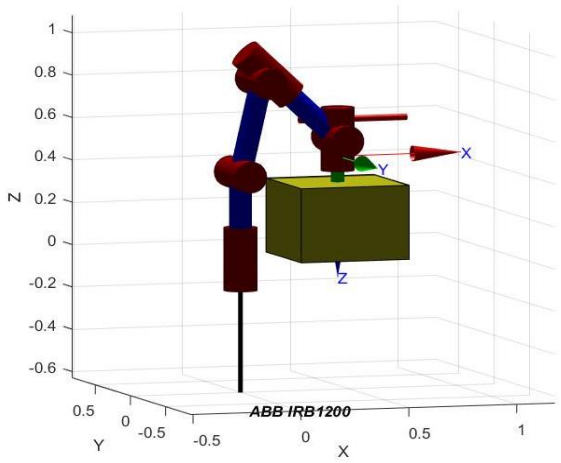

Fig 4. Required pose of the EF to grip the workpiece

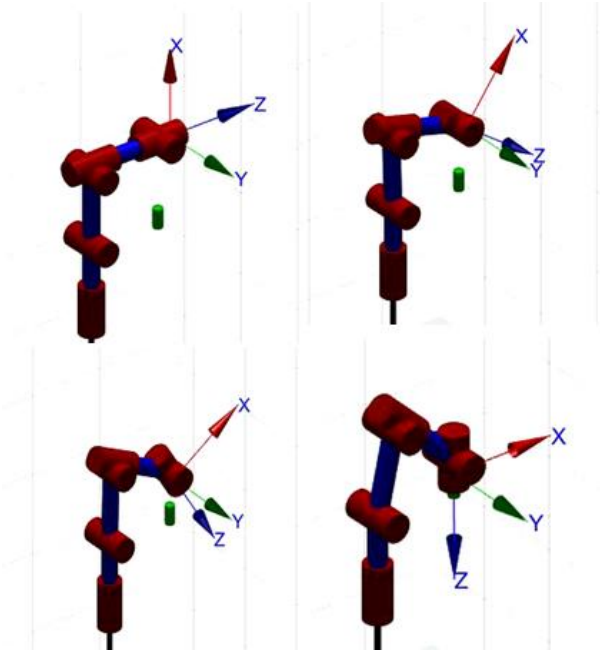

Fig 5. The procedure of picking the workpiece
Order : (1,1)→(1,2)→(2,1)→(2,2)

So, when we provided the velocity input, next q is calculated as (8). We did not consider the filter in this case because we did not differentiate the quantized signals yet.

$$q_{n+1} = q_n + v * 0.001 \tag{8}$$

In the simulation results, we confirmed that every joint angles and the EF tracked the desired trajectory well. It can be shown as Fig.6. We attached only three joint angles' trajectories as an example. In this case, only 2, 3, 5 joint angles changed and others are constantly zero. While other tasks are proceeded later, we compared the real trajectory with reference trajectory (required to accomplish the task) about 2, 3, 5 joints for convenience. You can find the every joint angle's graph in the MATLAB code.

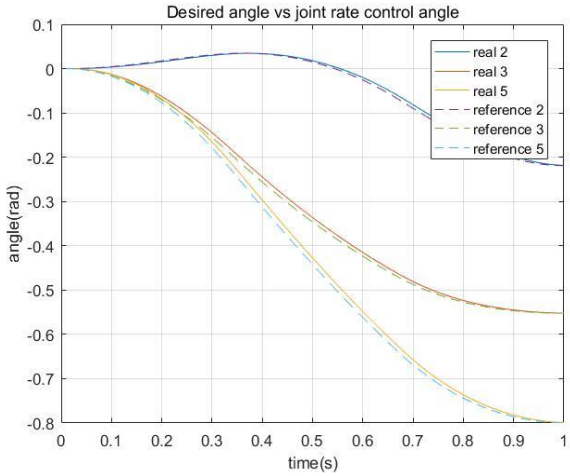

Fig 6. Joint rate control trajectory comparing with reference

III. Removing Rust

In Section II, the manipulator succeeded in gripping the workpiece. Thus, the manipulator needs to move the workpiece in front of the laser and move back and forth once while rotating the workpiece in a whole 360 degrees. We divided it into two procedures. The first procedure is moving the workpiece toward the desired position and orientation. It can be solved as same as 'Picking up'. The second procedure is moving the workpiece back and forth in y-direction of fixed frame while rotating. We updated the angle rate in 1kHz and simulated it in 3 seconds. (Positioning 1s, moving backward 1s, forward 1s)

A. The required Pose of the end-effector

In the first procedure, since the required distance between the surface and the laser is 0.05, the center of workpiece is 0.08 from the laser. So, we want to get the desired transformation matrix, which has a position vector [0.320;0.400;0.600] and a rotation matrix [0 1 0;0 0 1;1 0 0]. In this case, z-direction in EF frame should coincide with y-direction in fixed frame. To do so, we set the rotation matrix as above. We easily obtained the inverse kinematics using *ikine* function as same as above. In the second procedure, we found the trajectory from $g_{EF_{laser}}$ to $g_{EF_{laser}} * translate(0,0,-0.12)$with *ctraj* function. It means that moving backward by the cylinder's height. When we want to move forward, we tracked the trajectory of moving backward reversely.

$$g_{EF_{laser}} = \begin{bmatrix} 0 & 1 & 0 & 0.32 \\ 0 & 0 & 1 & 0.40 \\ 1 & 0 & 0 & 0.60 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$g_{EF_{remove}} = g_{EF_{laser}} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

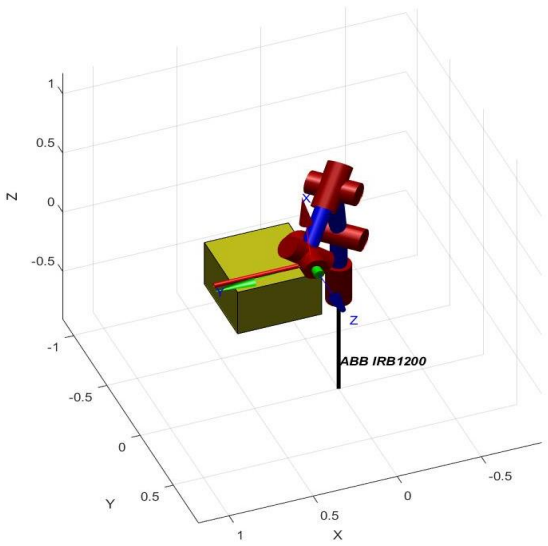

Fig 7. Required pose of the EF in front of the laser (red line :laser)

B. *Joint angle controller*

We can offer the joint angle directly and we already know the trajectory to remove the rust as above. We just added rotating motions to joint 6 while the workpiece moves forward and backward. It is enough to add 2π into the joint 6's angle when the workpiece moves backward. When the workpiece moves backward, we can subtract 2π from the joint 6's angle. In Fig.7, we denote a laser as a long red line. As mentioned above, it seems that laser touches the joint. However, it is not matter of inaccurate position. It is because the default size of the joint is too large.

C. *Joint rate controller*

The process of designing this controller is similar to picking up joint rate controller. We used the inverse kinematics trajectory with adding rotational motions while moving forward and backward as the desired trajectory. As a picking up joint rate controller, we designed a feedback controller with P control (k=100) as (12). Also, we can calculate the next q with 1 KHz updates as (13). Although we can complement state errors with I control, we do not need to consider some disturbances because no friction is assumed.

$$\dot{q} = v \tag{11}$$

$$v = -k(q - q_d) \tag{12}$$

$$q_{n+1} = q_n + v * 0.001 \tag{13}$$

In the simulation results, we confirmed that every joint angles and the EF tracked the desired trajectory well. It can be shown as Fig.8. ~Fig.10. We attached only three joint angles' trajectories as an example.

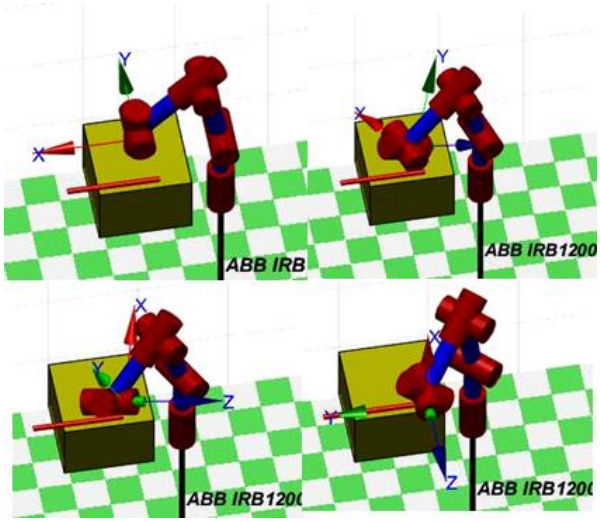

Fig 8. The procedure of moving the workpiece in front of the laser (Order : (1,1)→(1,2)→(2,1)→(2,2))

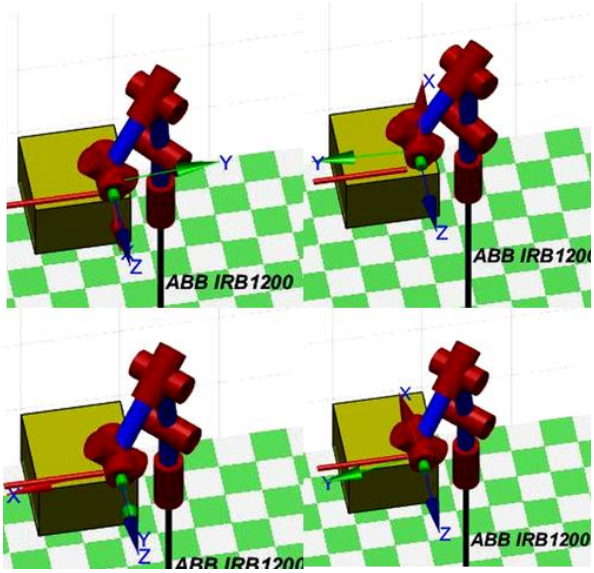

Fig 9. The procedure of moving the workpiece backward and forward (Order : (1,1)→(1,2)→(2,1)→(2,2))

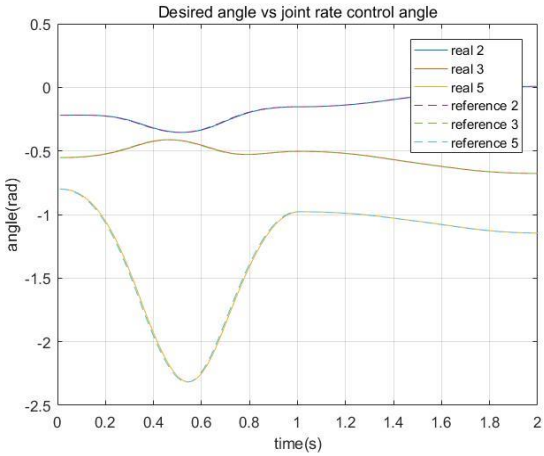

Fig 10. Joint rate control trajectory comparing with reference

IV. 3-DOF ROBOT DYNAMICS

We obtained the 3-DOF dynamics from Euler-Lagrange equation. The structure of this equation of robot is described as (14). We can calculate this *M, C, g* as the formula from Spong[1] such as (15) ~ (19). The whole process is included in code name *dynamics_withload*.

$$M\ddot{q} + C\dot{q} + g(q) = \tau + J^T F_{ext} \tag{14}$$

$$M = \sum_1^3 m_i J_{v_i}^T J_{v_i} + J_{w_i}^T R_i I_i R_i^T J_{w_i} \tag{15}$$

$$C = \sum_{i,j} c_{ijk}\dot{q}_i \ , c_{ijk} = \frac{1}{2}\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\} \tag{16}$$

$$g = \frac{\partial P}{\partial q_k} \ (P: potential\ energy) \tag{17}$$

$$p1 = \begin{pmatrix} 0 \\ 0 \\ L_{c1} \end{pmatrix} \ p2 = \begin{pmatrix} -L_{c2}\, sin(q_2)\, cos(q_1) \\ -L_{c2}\, sin(q_2)\, sin(q_1) \\ L_1 + L_{c2} cos(q_2) \end{pmatrix} \tag{18}$$

$$p_3 = \begin{pmatrix} -L_2\, sin(q_2)\, cos(q_1) \\ -L_2\, sin(q_2)\, sin(q_1) \\ L_1 + L_2 cos(q_2) \end{pmatrix} + \begin{pmatrix} -L_{c3}\, sin(q_2 + q_3)\, cos(q_1) \\ -L_{c3}\, sin(q_2 + q_3)\, sin(q_1) \\ L_{c3} cos(q_2 + q_3) \end{pmatrix} \tag{19}$$

In this case, it differs from original 3-DOF robot dynamics because it has a point mass at the distance of 0.451 from EF. We have two methods to consider this. First, we can include the point mass in the Link 3. Second, we can consider the point mass as the external wrench. If we assume that we do not know the value of the point mass, the first method is difficult because the Center Of Mass (COM) changes. Thus, we used the second method before we estimated the accurate mass. Because the full-derived dynamics are so complicated, it is attached in the MATLAB code.

V. ALGORITHM TO ESTIMATE THE MASS

We can only measure the joint angle at each moment from the encoder. From this measurement, we can differentiate the signals and obtain the angular velocity and angular acceleration. But, we cannot measure the torque or force. However, we can calculate the Jacobian and *M, C, g* from the measured angle, calculated angular velocity, and the angular acceleration. If we do not apply any input torque, $J^T F_{ext}$ can be calculated with only angle measurement from (20). Also, external wrench is described as below. We denote the unknown point mass as m_{un}

$$\mathrm{F_{ext}} = \begin{bmatrix} 0 \\ 0 \\ -m_{un} g \\ m_{un} lg * cos(q_2 + q_3) \sin(q_1) \\ -m_{un} l * g cos(q_2 + q_3) \cos(q_1) \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ -g \\ lg * cos(q_2 + q_3) \sin(q_1) \\ -lg * cos(q_2 + q_3) \cos(q_1) \\ 0 \end{bmatrix} m_{un} = f(q) * m_{un} \tag{20}$$

So, we can estimate m_{un} through least square method as (21). When we get additional data about $J^T f$ and $M\ddot{q} + C\dot{q} + g(q)$, we can add this 3×1 vector under the previous column. It means that we have $3\mathrm{n} \times 1$ vector if we have n data.

$$\min_{m_{un}} \| J^T f * m_{un} - M\ddot{q} + C\dot{q} + g(q) \| \tag{21}$$

This least square method can be solved as a minimization of quadratic function. So, m_{un} can be estimated from the solution of (21) to be,

$$((J^T f)^T (J^T f))^{-1} \ (J^T f)^T \big(M\ddot{q} + C\dot{q} + g(q)\big) \tag{22}$$

Of course it is convenient not to apply any force. However, if m_{un} is large enough to create fast angular velocity, there can exist large errors in estimation of velocity from the encoder. So, we need to prevent overly large velocity with applying appropriate torque input. In fact, we know the mass of link4, 5, 6 and do not know only the mass of the workpiece, which is usually much lighter than the robot manipulator. Therefore, it can be assumed that we know the approximate value range of the point mass. We can apply the torque to lower the speed and increase the accuracy. Also, we can know the applied torque and we can revise the least square method like (23). It can be also calculated from only angle measurement and torque value we already knew.

$$\min_{m_{un}} \| J^T f * m_{un} - (M\ddot{q} + C\dot{q} + g(q) - \tau) \| \tag{23}$$

In the simulation, we used S-function to simulate robot dynamics as close as reality. It provides the more accurate solution by solving ODE in various ways. We assumed m_{un} is 21kg, which consists of 19kg of link4, link5, and link6 and 2kg of workpiece. We applied input torque which can offset the wrench of 19kg. After that, we measured the joint angle and calculated the required parameters. It is contained in *Algorithm_estimate_mass* m.file and *find_b* function. We used 300 data to estimate the mass. In this simulation, estimated mass is about 21.13kg.

VI. JOINT TORQUE CONTROL

A. Picking up & Removing rust controller design

In this joint torque controller, we tried to accomplish the two tasks at the same time. If we already knew the q_d from Sec. II and Sec. III, we can simulate the dynamics with torque controller through Simulink S-function. In order to tackle the problem easily, let's change the dynamics. In Sec. V, we already knew the point mass. So, we included the point mass into the link 3's dynamical parameters. It means that we derived

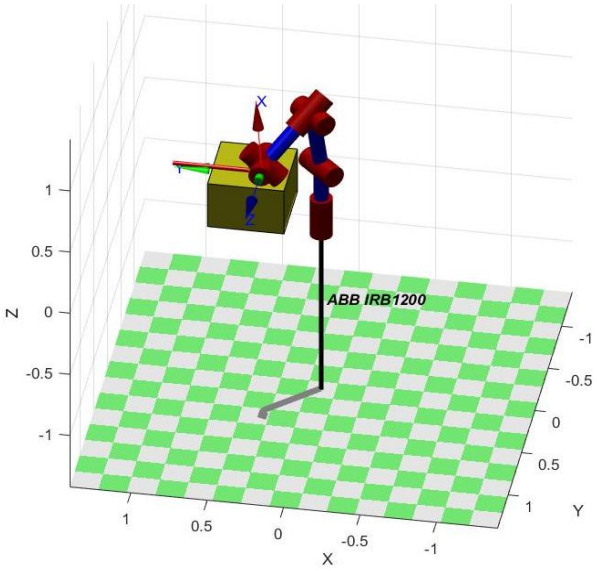

Fig 11. The required pose from the torque control

the dynamics from the first method in Sec IV. New derived dynamics are described as (24). In this dynamics, $J^T F_{ext}$ disappeared and it was included in M_r, C_r, g_r. Based on this dynamics, we designed the joint torque controller as below. We used feedback linearization (Computed tourqe control) to track the desired trajectory.

$$M_r\ddot{q} + C_r\dot{q} + g_r(q) = \tau \tag{24}$$

$$\tau = M\left(\ddot{q}_d - K_d(\dot{q} - \dot{q}_d) - K_p(q - q_d)\right) + C\dot{q} + g(q) \tag{25}$$

We simulated the dynamics with torque control in Simulink S-function and the calculated joint angles with torque control were used to plot the graphic rendering in Robotic Toolbox. We obtained $\theta_1, \theta_2, \theta_3$ from dynamics and set $\theta_4, \theta_5, \theta_6$ same as the desired trajectory because joint 4, 5, 6 can be controlled by angle. We set the K_d, K_p gain that can generate critically damped motion by PD control as below. This is for the fast convergence on the desired angle.

$$K_p = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \quad K_d = \begin{bmatrix} 2500 & 0 & 0 \\ 0 & 2500 & 0 \\ 0 & 0 & 2500 \end{bmatrix} \tag{26}$$

Also, we should consider the quantization signals because we need velocity input for torque control. Quantization cannot be assumed that it is Gaussian noise. So, we just tried to smooth the signal's shape. If we smooth the discontinuous part, we can prevent the sudden increase of velocity. We chose the Low Pass Filter(LPF) to filter the signals before numerically differentiation. We made the user-defined function named LPF. LPF can be described simply as next.

$$x_{filter,n} = \frac{\tau}{\tau + t_s} x_{filter,n-1} + \frac{t_s}{\tau + t_s} x_{original} \tag{27}$$

x_{filter}: $data\ after\ filtering$

t_s: $sampleTime$, τ: $time\ constant$

So, we filtered every data before numerically differentiation.

B. *Simulation Results*

In the simulation results, torque control works well. It is enough to track the given trajectory. It can be shown at Fig.12. However, there were still some differences from joint angle control or joint rate control.

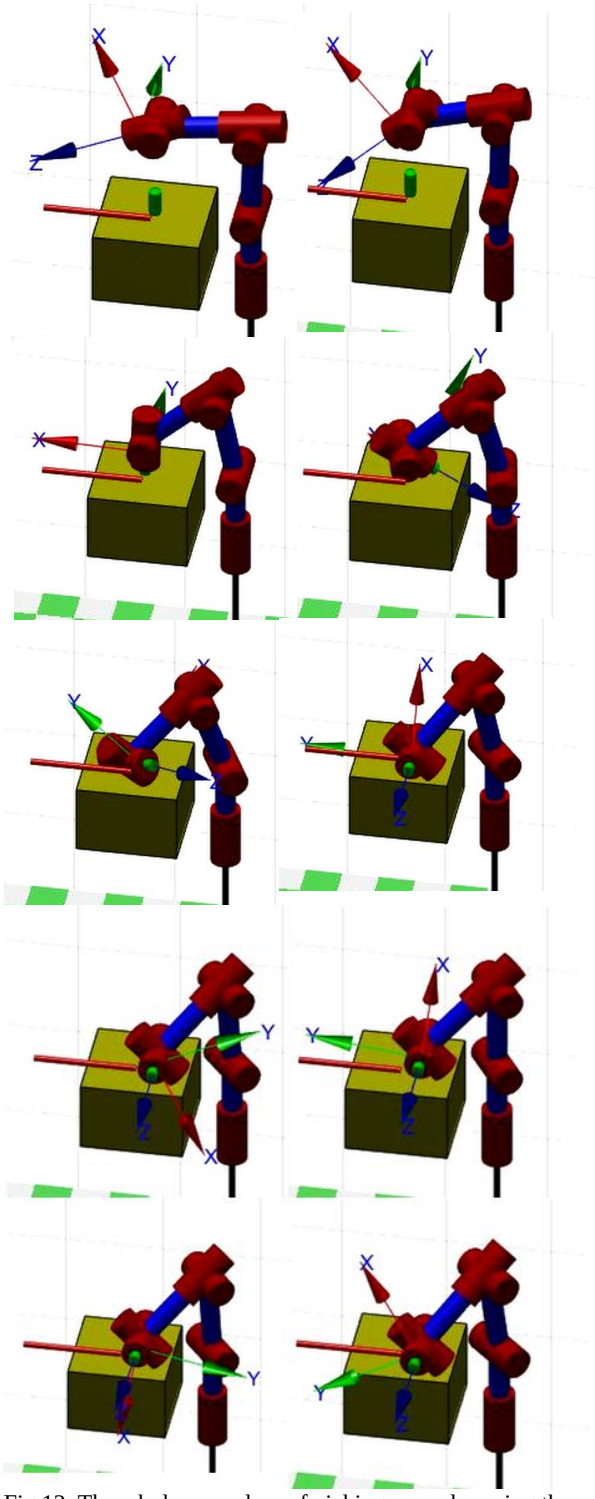

Fig 12. The whole procedure of picking up and moving the workpiece in front of the laser and moving backward and forward (Order : (1,1)→(1,2)→(2,1)→(2,2)→…→(5,2))

Although we set up the large PD gains which have critically damped features, some errors in some ranges were inevitable. You can see some errors in the beginning parts as Fig.13. If the smooth trajectory was given, fully-actuated systems can be controlled well by feedback linearization.

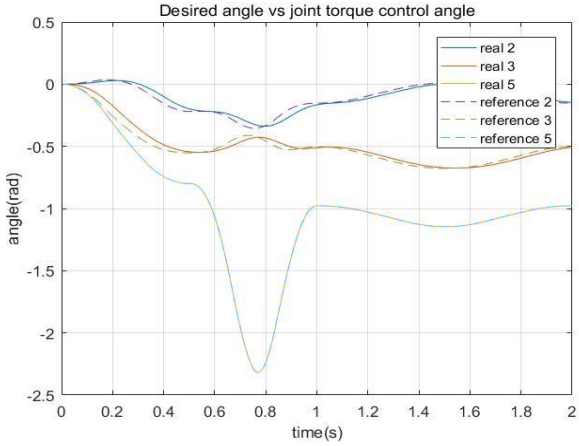

Fig 13. Joint torque control trajectory comparing with reference

VII. POLISHING TASK

To do this task, the workpiece should be located at [0.650;0.070;0.450] and EF should be located at [0.590; 0.070;0.450]. Also, orientation for polishing start is same as the initial pose of EF. Then, the required transformation is

$$g_{EF_{polish}} = \begin{bmatrix} 0 & 0 & 1 & 0.59 \\ 0 & -1 & 0 & 0.07 \\ 1 & 0 & 0 & 0.45 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (28)$$

This initial pose is described as Fig.14.

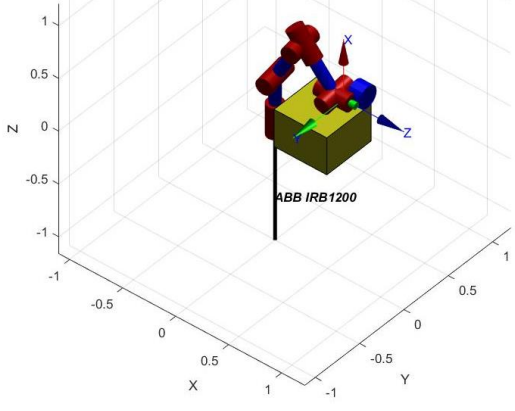

Fig 14. The initial pose with the pulley for sanding polishing.

This task should be based on hybrid position and force control. EF should be constrained on the circumference with +45dgree and -45dgree. Also, EF needs to create 5N force in the direction of radius. Sanding belt's position is [0.650;0.200;0.450], denoted as p_s. We also denote EF's position as [x;y;z] expressed in the pulley's middle point. nd pulley's radius as r_s. Then, holonomic constraint can be described as next.

$$x^2 + y^2 + z^2 = r_s^2 \quad (29)$$

(29) can be parameterized θ and ϕ as (30)

$$q_p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_s cos\theta cos\phi \\ r_s cos\theta sin\phi \\ r_s sin\theta \end{pmatrix} \quad (30)$$

$$\dot{q_p} = \begin{pmatrix} -r_s cos\theta sin\phi & -r_s sin\theta cos\phi \\ r_s cos\theta cos\phi & -r_s sin\theta sin\phi \\ 0 & r_s cos\theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \end{pmatrix} = J \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \end{pmatrix} \quad (31)$$

Then, we need to change the original dynamics to q_p dynamics. Forward kinematics can be considered as function and position vector of $g_{03}(\theta_1, \theta_2, \theta_3)$ is $h(q)$. Then, next relation is established.

$$h(q) = q_p + p_s \quad (32)$$

Then, we could find the next relationship.

$$\frac{\partial h}{\partial q}\dot{q} = \dot{q}_p \;\rightarrow\; \dot{q} = \left(\frac{\partial h}{\partial q}\right)^{-1} \dot{q}_p = J_q(q)\dot{q}_p \quad (33)$$

$$\ddot{q} = J_q(\dot{q})\dot{q}_p + J_q(q)\ddot{q}_p \quad (34)$$

Substitute (33), (34) into (24)

$$M_r\{J_q(\dot{q})\dot{q}_p + J_q(q)\ddot{q}_p\} + C_r J_q(q)\dot{q}_p + g_r(q) = \tau \quad (35)$$

We can arrange (35) and define new M_p, C_p, g_p.

$$M_p\ddot{q}_p + C_p\dot{q}_p + g_p(q_p) = \tau_p \quad (36)$$

When the constraints exists,

$$M_p\ddot{q}_p + C_p\dot{q}_p + g_p(q_p) + A^T\lambda = \tau_p \quad (37)$$

(30), (31) also can be substituted into (37), then we get

$$D\ddot{\phi} + Q\dot{\phi} + g_\phi(\phi) = \tau_\phi \quad (38)$$
$$D = J^T M_p J, Q = J^T(M_p\dot{J} + C_p J)$$

In this case, we can apply the hybrid position-force control as (39), if we have λ_d and ϕ_d.

$$\tau = MJ\left(\ddot{\phi}_d - K_v\dot{e}_\phi - K_p e_\phi\right) + \left[CJ + M\dot{J}\right]\dot{\phi} + g - f + A^T(\lambda_d - K_f\int(\lambda - \lambda_d)dt) \quad (39)$$

REFERENCES

[1] M.W. Spong, S. Hutchinson, and M. Vidyasagar, “Robot Modeling and control”, WILEY, 2006.