ME5402/EE5106 ADVANCED ROBOTICS

RRR Surgery Robot Controlling and Modelling

Matric No.	Group Member
A0260014Y	Shiping Guo
A0263721J	Yukai Wang
A0267500L	Ziyue Qin

Contents

Honor Pledge

ME5402/EE5106 ADVANCED ROBOTICS
EE5064 Dynamics and Control of Robot Manipulators

Group Projects

Year	Semester	Project Choice
2021-2022	**2**	**Project 3**

We, the following students of NUS, upon our honor, hereby confirm that we have neither received nor given any unauthorized help on the ME5402/EE5106 group project carried out by us.
The project reports reflect truly our own efforts. In all cases where material from other sources such as books, articles, notes and websites have been used, we have taken care to provide clear and unambiguous references to the same.
We confirm that we have not provided unauthorized help to other groups doing the same project. Furthermore, we also confirm that we will not pass on our research materials, report and presentation materials to other students who may take this module in later semesters.
In addition, this project report has been prepared and submitted by us only as a part of an academic exercise. Its contents are not meant for publication in any manner.

Signature table

Matric No.	No.	Name	Task	Signature	Date
A0260014Y	A	Shiping Guo	Task 3	Guo Shiping	22/4/2023
A0263721J	B	yukai wang	Task 4	Yukai Wang	22/4/2023
A0267500L	C	Ziyue Qin	Task 5	Qin Ziyue	22/4/2023

Contribution table

Tasks	Tasks weightage	Work distribution
Task 1	10%	All
Task 2	20%	All
Task 3	40%	A
Task 4	40%	B
Task 5	40%	C
Task 6	20%	All
Task 7	10%	All

List of figures

Chapter 1 Introduction and literature review

Arm robots have applications in manufacturing, healthcare, and agriculture. They can perform complex tasks with high precision and accuracy, improving efficiency and reducing labour costs. In article [1] dual-arm robot can perform tasks in place of human workers. In article [2] they designed a kind of multiple joints robot combined with computer version to help older Adults. In Figure 1-1, industrial and medical robotics have collaborated to implement one kind of surgical robot [3] has more movement options for the surgeons to operate in the human body[4] . In figure 1 left, it shows a model used robot arm assisted technique to play the role of computer-assisted surgery (CAS).

Figure 1 Surgical robot with more movement

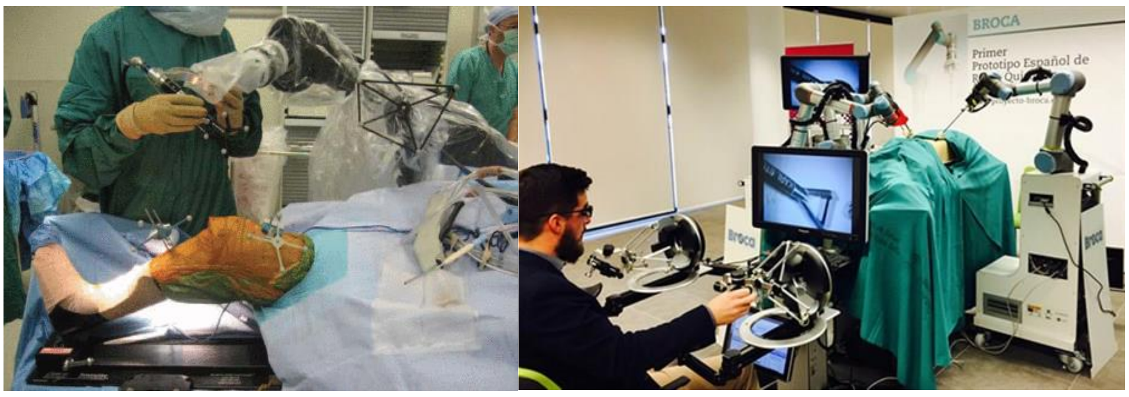


Figure 2 Computer-assisted surgery

Figure 2 right part shows the Universal Robots (UR5) robot assist doctor to do surgery which is very meaningful for remote surgery. UR5 is a highly versatile robotic arm designed for a wide range of applications in industries such as manufacturing, automotive, and healthcare. The importance of this robot lies in its ability to perform complex tasks with a high degree of accuracy and repeatability, while also offering flexibility in terms of programming and operation. The need for such a robot is evident in industries where repetitive or hazardous tasks are involved, as the UR5 can take over these tasks and improve efficiency and safety in the workplace. Additionally, the robot can work alongside human workers, allowing for collaborative processes that improve overall productivity just like computer-assisted surgery system.

Chapter 2 Robot design and applications

Designing a better robot arm and control system to meet the demanding requirements of surgery is a challenging and important topic. In this article, we simulated a simplified robot arm based on UR5 using classical control methods and compared the results to gain a better understanding of the underlying theory. This knowledge will aid us in designing more advanced and efficient robot arms and control systems in the future.

The UR5 robot arm is similar to other 3 degree-of-freedom robotic arms in its ability to perform tasks such as pick-and-place operations, machine tending, and assembly. In this article, we propose a simplified version of the UR5 by retaining the original three joints (as shown in Figure 2) to validate the theory of Forward Kinematics, Inverse Forward Kinematics, Jacobian, Newton Euler, Lagrange Euler, PID, and computed torque by simulation. This approach enables us to validate and improve the control system, which is critical for surgical robots. Figure 3 depicts our robot arm model. All the parameters we used in this article are based on the UR5 user manual [5]. The table1 shows some key parameters useful for us in this project.

Table 1 The dimension of UR5e robotic arm

Link#	Dimension	Value [mm)	Mass[Kg]	Center of Mass[mm]
1	d_1	89	3.7	[0, -25.61, 1.93]
2	a_2	425	8.393	[212.5, 0, 113.36]
3	a_3	392	2.33	[150, 0, 26.5]

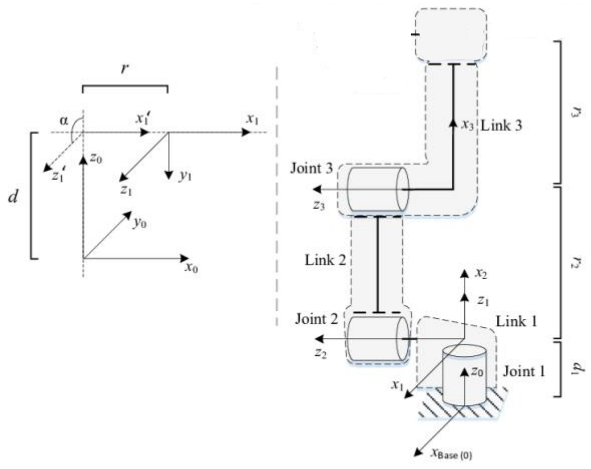


Figure 3 Configuration of Simplified UR5e and its dimensions.

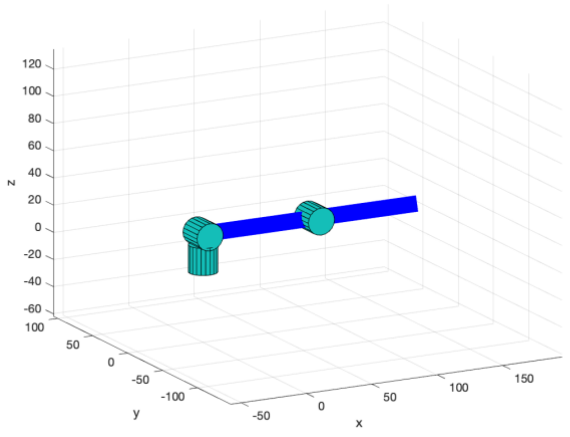


Figure 4 Simplified 3 DOF of UR5 model

Chapter 3 Kinematics and computing

3.1 Forward Kinematics analysis

Modelling of the robotic arm is essential for controlling it to achieve expected tasks. Additionally, numerical modelling can be easily transformed to computer-based simulation which is critical for this project. The first step to establish robotic model is forward kinematics.

It is a function that takes the angle of each joint as input and outputs the position and pose of end effector in space. In order to achieve this function, the relevant coordinate of each joint and dimension of the robotic arm ought to be discovered first. In Figure 5, the cartesian coordinates and joint displacement of each joint are represented in $x_i, y_i, z_i$and θ_i , where i means the i-th joint and i could take 1 to 3 separately. The dimension of each link of robotic arm are listed in Table 1.

Denavit-Hartenberg (D-H) representation is a systematic method to representing kinematic relationship between two adjacent links of robotic arm. The D-H representation contains four parameters associated with joint i, d_i, a_i, α_i, and θ_i. And each of the D-H parameters are associated with link relationship between link i and link i-1, the D-H table is shown in table 2.

Parameter d denotes the distance from x_i axis to x_i axis along the z_{i-1} direction, a means distance between common normal axis, α represents link twist angle difference between z_i axis and z_{i+1}axis, while the θ_i is the joint angle difference between x_{i-1} axis and x_i axis. Therefore, the D-H representation can be listed in Table 2 with 3 revolute joints, where value of d and a are defined in Table 1 and value of θ is subject to change during robotic manipulation. Once the D-H table is established, the transformation matrix between any pair of adjacent joints,

Table 2. D-H representation table.

Joint i	$\boldsymbol{d_i}$**[mm]**	$\boldsymbol{a_i}$**[mm]**	α_i**[rad]**	$\boldsymbol{\theta_i}$**[rad]**
1	d_1	0	$\pi/2$	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3

A_i^{i-1} can be found based on (Eq.1). Meanwhile, by multiplying transformation matrix of each pair links all together (Eq.2), the total transformation matrix T_i^0 between base coordinate and end-effector can be found which is the kinematic equation of the robotic arm. In another word, the spatial position and rotational angle of the end-effector now can be represented as a function of joint displacements θ_i and i takes value of 1 to 3.

$$A_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & \alpha_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & \alpha_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq.1)}$$

$$T_i^0 = A_1^0(\theta_1)A_2^1(\theta_2) \ldots A_i^{i-1}(\theta_i) \quad \text{(Eq.2)}$$

Transformation matrixes A_i^{i-1}(i takes value of 1 to 3) can be calculated when θ_i are all set to zero for demonstration purpose. Furthermore, the kinematic equation result of the robotic arm can also be illustrated. By inputting $\theta_i = [0, \frac{pi}{2}, 0]$ we get the result as shown in Figure 7. Compared with the initial position in figure 6, the second joint turns 90°. It also approves the correction of our UI design. In the following parts, all the visualization parts are based on this robot model UI.

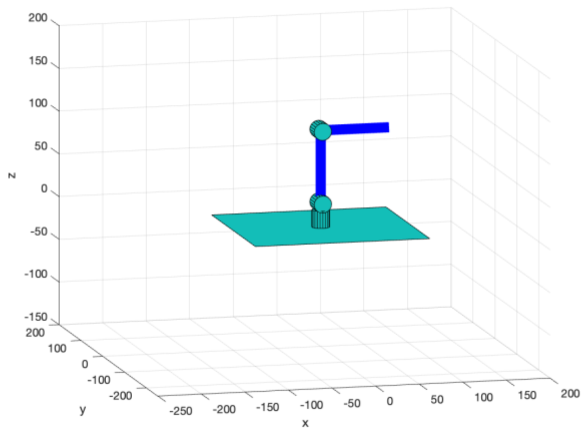


Figure 5 Forwards control validation

3.2 Forward Kinematics MATLAB simulation

This part mainly shows the simulation process in MATLAB coding. The figures below show computational process of implementing a forward kinematic function in MATLAB.

```
function [T_01, T_02, T_end] = Forward(theta)
% DH table
   %          a      alfa         d           theta
DH=[       0,      pi/2,       0.0892,     theta(1);
        0.425,            0,       0.0,           theta(2);
        0.392,            0,       0.0,          theta(3)];
% Calculate the T one by one
T_01 = (Translation(DH(1,3), 'z') * Rotation(DH(1,4), 'z') * Translation(DH(1,1), 'x') * Rotation(DH(1,2),
'x'));
T_12 = (Translation(DH(2,3), 'z') * Rotation(DH(2,4), 'z') * Translation(DH(2,1), 'x') * Rotation(DH(2,2),
'x'));
T_23 = (Translation(DH(3,3), 'z') * Rotation(DH(3,4), 'z') * Translation(DH(3,1), 'x') * Rotation(DH(3,2),
'x'));
%Calculate the T 0-6
T_02 = (T_01 * T_12);
T_end = (T_02 * T_23);
end
```

The function called Translation and Rotation is derived by the following computing process.

```
function [T] = Translation(d, ax)
   T = (eye(4));
   if (ax == 'x')
      T(1,4) =+ d;
   elseif (ax == 'z')
      T(3,4) =+ d;
   end
end

function [R] = Rotation(r, ax)
   R = (eye(4));
   rot = ([cos(r),-sin(r);sin(r),cos(r)]);
   if (ax == 'x')
      R(2:3,2:3) = rot;
   elseif (ax == 'z')
      R(1:2,1:2) = rot;
   end
end
```

After computing each transformation matrix, the corresponding T_i^{i-1} can be derived. Then the position information of each joint on robotic arm can be found in their corresponding T_i^{i-1} matrix.

3.3 Inverse Kinematics analysis

The geometric approach method is a viable option for solving the inverse kinematic problem. This method involves using the cosine rule of the angle in a triangle, as well as Pythagoras' law and trigonometric rules, to determine the desired end-effector position angle. This approach has been adapted from a model presented in [6], where a small-scale 3 DoF arm robot was constructed for pick and place missions. To implement the method, each link on the robotic arm is shaped like a triangle, as shown in Figure 3-2. The solution for the inverse kinematic model is derived from the following equations.

a) The solution to find the base joint angle θ_1

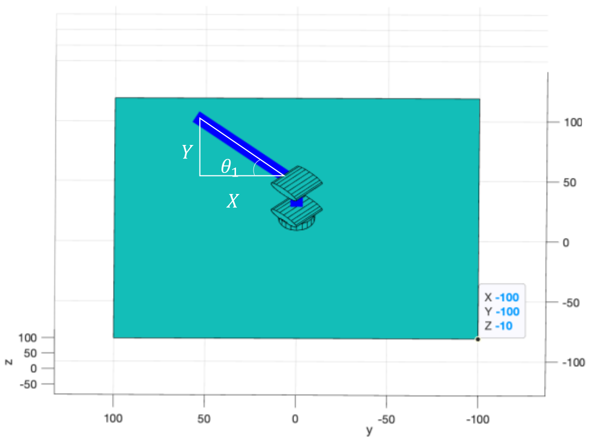


Figure 6 The base joint angle

If X > 0, then :

$$\theta_1 = tan^{-1}[\frac{Yef}{Xef}] \qquad \text{(Eq.3)}$$

If X < 0, then :

$$\theta_1 = 180° - tan^{1}[\frac{Yef}{Xef}] \qquad \text{(Eq.4)}$$

b) Find the second joint angle (θ_2)

Find r_1, r_2, and r_3:

$$r_1 = \sqrt{Xef^2 + Yef^2} \quad \text{(Eq.5)}$$

$$r_2 = Zef - a_1 \quad \text{(Eq.6)}$$

$$r_3 = \sqrt{{r_2}^2 + {r_1}^2} \quad \text{(Eq.7)}$$

Find j_1 and j_2 :

$$j_1 = tan^{-1}[\frac{r_2}{r_1}] \quad \text{(Eq.8)}$$

$$j_2 = cos^{-1}[\frac{{a_2}^2+{r_3}^2-{a_3}^2}{2a_2r_3}] \quad \text{(Eq.9)}$$

So the value of θ_2 :

$$\theta_2 = j_1 + j_2 \quad \text{(Eq.10)}$$

c) Find third joint angle θ_3

$$j_2 = cos^{-1}[\frac{{a_2}^2+{r_3}^2-{a_3}^2}{2a_2r_3}] \quad \text{(Eq.11)}$$

$$\theta_3 = -(180° - j_3) \quad \text{(Eq.12)}$$

3.4 Inverse Kinematics MATLAB simulation

In MATLAB, “fsolve” function is a great tool to solve non-linear problem, especially for inverse kinematic calculate of robot arm. In our codes, we make good use of this function and calculate the result easily as shown below.

Key codes(Part)

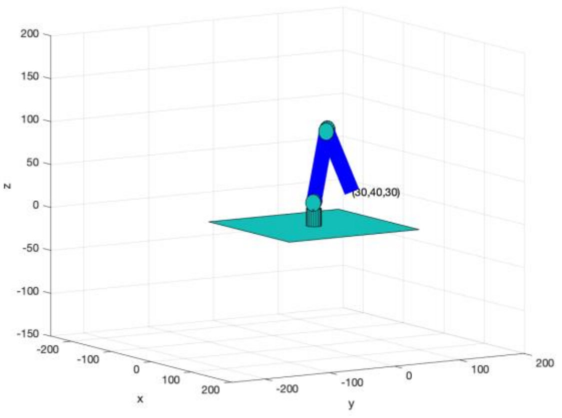
```
% Solve for theta2&3
% Solve for theta2
beta = atan2((z_bar-d1),(L));
gamma = acos((L^2+(z_bar-d1)^2+a2^2-a3^2)/(2*a2*sqrt((L)^2+(z_bar-d1)^2)));
theta2 = gamma+beta;

% Solve for theta3
theta3 = fsolve(@(theta)[cos(theta); sin(theta)]-(1/(a3^2))*[a3*cos(theta2), a3*sin(theta2); -
a3*sin(theta2), a3*cos(theta2)]*[L-a2*cos(theta2);z_bar-a2*sin(theta2)-d1],0);

% Solve for theta1
```

```
theta1 = fsolve(@(theta) [cos(theta); sin(theta)]-(1/(x_bar^2+y_bar^2))*[x_bar, y_bar; y_bar, -
x_bar]*[a3*cos(theta3)*cos(theta2)+a2*cos(theta2)-a3*sin(theta2)*sin(theta3); 0], 0);
theta = [real(theta1), real(theta2),real(theta3)];
```

Check the inverse kinematic calculation:

```
>>Locus(30,40,30,0,0)   % give the target position
>>0.9273    1.3693   -2.5058  % get rotation angle of the three torques
```

By using the calculated torque θ_1 , θ_2 , *and* θ_3 to control the robot arm. The end effector position is equal to the original input as shown in Figure 3-3.



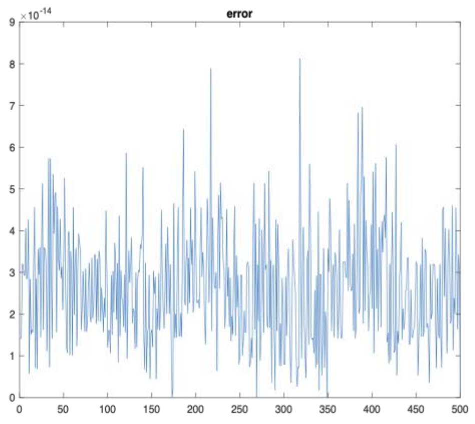


Figure 7 Inverse Kinematic validation   *Figure 8 The error between IK result and original input*

All the errors are due to the reduction in the calculation, the system has no noise, so the overall error is very small as shown in Figure 10, which also proves that our derivation is correct**.**

3.5 Jacobian Matrix

In order to achieve the task that assign particular speed for robotic arm to draw trajectory, the Jacobian Matrix is required to be analysed. If the speed of end-effector is known, the velocity vector of each joints can be calculated based on the formular given below.

$$\dot{x} = J\dot{q} \tag{Eq.13}$$

where x is position and orientation of end-effector in Cartesian coordinate, while q is the position of each joint. Be taking derivation of these x and q, a linear relationship, or the Jacobian Matrix J between rotational rate of joints and speed of end-effector can be defined.

Jacobian matrix can be used to describe the relationship between velocity and acceleration of the end-effector in the robot arm and of individuals joints. Thus, it can control the movement of the robot arm. Jacobian matrix is consisting of vectors representing translational and rotational movement of each joint of robotic arm, as shown below.

$$J = \begin{bmatrix} J_{L_1} & \cdots & J_{L_i} \\ J_{A_1} & \dots & J_{A_i} \end{bmatrix} \tag{Eq.14}$$

where J_L is linear velocity component of each joint and J_A is angular velocity component of each joint, while i is the number of joints that robotic arm has. For this project, the robotic arm has six revolute joints. The Jacobian matrix can be further written as:

$$J_i = \begin{bmatrix} J_{L_I} \\ J_{A_1} \end{bmatrix} = \begin{bmatrix} b_{i-1} \times r_{i-1,e} \\ b_{i-1} \end{bmatrix} \qquad \text{(Eq.15}$$

where b_{i-1} is the unit vector along z-axis of frame i-1, and $r_{i-1,e}$ is the position vector from frame $i-1$ to end-effector.

$$J_A = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$J_A = \begin{bmatrix} 0 & s_1 & s_1 & s & c_1 s & r_{13} \\ 0 & -c_1 & -c_1 & -c_1 & s_1 s_{234} & r_{23} \\ 1 & 0 & 0 & 0 & -c_{234} & r_{33} \end{bmatrix} \tag{Eq.16}$$

In Equation 16, the Jacobian matrix for revolute joints can be easily found by identifying b_{i-1} the unit vector representing the z-axis of joint $i-1$ relative to base frame. And for the purpose of convenience, s_1 means cousin of joint angle 1, s_{23} means sine of sum of joint angle 2, 3. Furthermore, the Jacobian matrix for prismatic joints can be found by identifying $r_{i-1,e}$, the relative position with respect to frame i-1 and multiply them with relative b_{i-1} :

$$r_{0,e} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \qquad \text{(Eq.17)}$$

$$r_{1,e} = \begin{bmatrix} p_x \\ p_y \\ p_z - d_1 \end{bmatrix} \qquad \text{(Eq.18)}$$

$$r_{2,e} = \begin{bmatrix} p_x - c_1 c_2 a_1 \\ p_y - c_2 s_1 a_2 \\ p_z - s_2 a_2 - d_1 \end{bmatrix} \qquad \text{(Eq.19)}$$

$$J_L = \begin{bmatrix} -p_y & -c_1 p_z + c_1 d_1 & c_1(s_{234} s_5 d_6 + c_{234} d_5 - s_{23} a_3) \\ p_x & -s_1 p_z + s_1 d_1 & s_1(s_{234} s_5 d_6 + c_{234} d_5 - s_{23} a_3) \\ 0 & -s_1 p_z + s_1 d_1 & -c_{234} s_5 d_6 + c_{234} d_5 + c_{23} a_3 \end{bmatrix} \quad \text{(Eq.33)}$$

Besides, the numerical method also can be applied to calculate Jacobian matrix. In this simulation, both methods are tried. Hence, a numerical solution method called finite differences is introduced. For finite differences method, through adding small increments, the translation change and rotation change of the robot arm end effector can be calculated in each individual joints.

$$Increment = 1e - 6 \qquad \text{(Eq.20)}$$

$$T_{new} = Forwardkinematics(a, alpha, d, theta + Increment) \quad \text{(Eq.21)}$$

$$J_{L(:,i)} = \left[T_{new(1:3,4)} - T_{(1:3,4)}\right] / Increment \qquad \text{(Eq.22)}$$

$$J_{A(:,i)} = \left[T_{new(1:3,1:3)} - T_{(1:3,1:3)}\right] / Increment \qquad \text{(Eq.23)}$$

Finally, the difference is divided by the small increment to obtain the partial differential values of position and attitude so that the Jacobian matrix can be calculated.

3.6 Jacobian Matrix MATLAB Simulation

This part mainly shows the simulation process in MATLAB coding. The codes below show computational process of implementing Jacobian calculation function in MATLAB which just follow the instruction in 3.5 Jacobian Matrix and use .

```
function [T_01, T_02, T_end, J] = Jacobian_calculation(q1, q2, q3)
% This function is used to calculate the J
% First to get the T
[T_01, T_02, T_end] = Forward([q1, q2, q3]);
p_eff = T_end(1:3,4);     % The end position
J = (zeros(6));
z0 = [0;0;1];
% To calculate the position from 0-3 one by one
J(1:3,1) = cross(z0, p_eff - z0);
J(1:3,2) = cross(T_01(1:3,3), p_eff - T_01(1:3,4));
J(1:3,3) = cross(T_02(1:3,3), p_eff - T_02(1:3,4));
% To calculate the orienation from 0-3 one by one
J(4:6,1) = z0;
J(4:6,2) = T_01(1:3,3);
J(4:6,3) = T_02(1:3,3);
end
```

Later we calculate the inverse Jacobian in two ways, the first way is directly use pinv function to calculate inverse Jacobian.

```
Inverse Jacobian  = pinv(J_curr)
```

The second method is:

```
Winv = zeros(6);
for i = 1:6
   Winv(i,i) = i/6 + 0.1;
end
Inverse Jacobian  = Winv * J_curr' * inv(J_curr * Winv * J_curr' + inv(C));
```

By comparation, the results by using these two methods are almost the same.

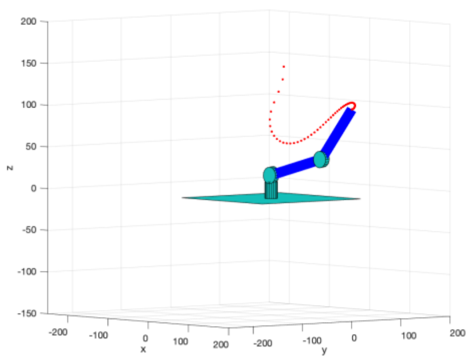


Figure 9 Control speed by using Jacobian method.

Jacobian method is a relatively simple but powerful tool for controlling the end-effector speed of a robotic arm which is crucial for surgery robot. without having to worry about the underlying joint movements. Just as shown in Figure 11, By adjusting the joint velocities, the method can generate a smooth and efficient trajectory that avoids obstacles and reduces the time required to complete a task. It can handle non-linearities and singularities in the robot's kinematic model, which can be difficult to deal with using other methods. This is particularly important for complex robotic systems, where non-linearities and singularities are common.

By the way, we also found some problems, singularity exists when we use high speed to control the robot. Under this circumstance, the robot arm is out of control and move randomly which is a fatal problem for surgical robots. There are many singularity analysis methods can solve this problem. For example, the paper[7] analyses a 3-degree of freedom spatial parallel robot to improve dynamic stability during manipulation tasks. It investigates singular points using a numerical solution and SolidWorks software, and identifies two singular points. The findings can be used to plan the mechanism in a non-singular way during object manipulation. There also some other methods to control the singularity problem, which is also our next goal to improvement it.

Chapter 4 Dynamics and Computing

Dynamics is a vast branch of study that focuses on the forces necessary to generate motion. In a robotic system, the dynamic motion of the manipulator arm is produced by the torques created by the actuators [10]. This relationship between the input torques and the time rates of change of the robot arm component configurations represents the dynamic modelling of the robotic system, which is concerned with the derivation of the manipulator's equations of motion as a function of the forces and moments acting on it [10]. As a result, dynamic modelling of a robot manipulator entails determining the mapping between forces applied on structures and joint locations, velocities, and accelerations. In the following section, two methods are used to derive dynamic equation of a 3 DOF rotational robotic arm. Newton-Euler method and Langrage-Euler method are derived in below.

4.1 Newton-Euler Formulation

The Newton-Euler formulation is derived from Newton's Second Law of Motion, which defines dynamic systems in terms of force and momentum [11]. The equations include all of the forces and moments operating on the individual robot linkages, as well as the coupling forces and moments. The Newton-Euler formulation is divided into two steps: forward and backward recursion [11]. The forward recursion begins from the robot's base and progresses to the end-effector, computing the acceleration, velocity, and location of each link. The forces and torques operating on each link are calculated using the equations of motion. Backward recursion, on the other hand, begins at the end-effector and works its way back to the base, computing the forces and torques necessary at each joint to accomplish the desired motion trajectory.

For outward iterations, angular displacement of each link (rotational) can be expressed as:

$$\omega_{i+1}^{i+1} = R_i^{i+1}\omega_i^i + \dot{\theta}_{i+1}Z_{i+1}^{i+1} \qquad \text{(Eq.24)}$$

where i is the number of links; R_i^{i+1} is the rotational matrix between link i+1 and link i; $\dot{\theta}_{i+1}$ is the time derivative of angle of link i+1, and Z_{i+1}^{i+1} is the rotational axis of link i+1, with

$$\dot{\theta}_{i+1}Z_{i+1}^{i+1} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}^{i+1} \qquad \text{(Eq.25)}$$

The angular velocity, $\dot{\omega}$ is then be derived as follows:

$$\dot{\omega}_{i+1}^{i+1} = R_i^{i+1}\dot{\omega}_i^i + \ddot{\theta}_{i+1}Z_{i+1}^{i+1} + R_i^{i+1}\omega_i^i\dot{\theta}_{i+1}^{i+1}Z_{i+1}^{i+1} \qquad \text{(Eq.26)}$$

where $\ddot{\theta}_{i+1}$ is the second time derivative of angle of link i+1. Additionally, as the first link of the robotic arm, the base, is not rotating. Thus, ω_1^1 and $\dot{\omega}_1^1$ equals all zero.

Next, the linear acceleration of the coordinate is shown as below:

$$\dot{v}_{i+1}^{i+1} = R_i^{i+1}[\dot{\omega}_i^i P_{i+1}^i + \omega_i^i(\omega_i^i P_{i+1}^i) + \dot{V}_i^i] \qquad \text{(Eq.27)}$$

where P_{i+1}^i is the position of link i coordinate with respect to link i+1 coordinate.

For inward iterations, the force and moment exerted on link I by link i-1 is as follows:

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} + F_{i+1}^{i+1} \qquad \text{(Eq.28)}$$

$$n_i^i = R_{i+1}^i n_{i+1}^{i+1} + P_{Ci}F_i^i + P_{i+1}^i R_{i+1}^i f_{i+1}^{i+1} + N_i^i \qquad \text{(Eq.29)}$$

where F_i=ma=m$\dot{v}_i$ represents net force exerted on link I, while P_{Ci} is the position of the centre of mass of link i.

Finally, the torque can be calculated with n and Z:

$$\tau_i = (n_i^i)^T Z_i^i \qquad \text{(Eq.30)}$$

For the simulation in this assignment, for the purpose of easy of calculation, the centre of mass of each link is set to be at the joint. Plus, several initial conditions are made as well:

$$z_0 = \begin{bmatrix}0\\0\\1\end{bmatrix}, \omega_0 = \begin{bmatrix}0\\0\\1\end{bmatrix}, \dot{\omega}_0 = \begin{bmatrix}0\\0\\1\end{bmatrix}, v_0 = \begin{bmatrix}0\\0\\0\end{bmatrix} \qquad \text{(Eq.31)}$$

Key codes

```
%% Derive Newton-Euler formulation -- 3DOF RRR manipulator
syms l1 l2 l3 m g I w0 dw0 dv0 c1 s1 c2 s2 c3 s3 dq dq1 dq2 dq3 dw Pc F N f n m1 m2 m3 ddq1 ddq2
ddq3
syms q0 dq0 ddq0
R=cell(1,6);w=cell(1,6);dw=cell(1,6);dq=cell(1,6);ddq=cell(1,6);dv=cell(1,6);
f=cell(1,6);n=cell(1,6);P=cell(1,6);Pc=cell(1,6);F=cell(1,6);m=cell(1,6);N=cell(1,6);
```

```
I=cell(1,6);

%Initial value
R{1}=[c1 -s1 0;s1 c1 0;0 0 1];R{2}=[c2 0 s2 ;0 1 0;-s2  0 c2 ];R{3}=[c3 0 s3 ;0 1 0;-s3  0 c3 ];
R{4} = [1 0 0;0 1 0; 0 0 1];

dv{1}=[0;g;0];
%Coordinate position, offset condition
P{1}=[0;0;0];P{2}=[0;0;0];P{3}=[l1;0;0];P{4}=[l2;0;0];
%the centre of mass of each link
Pc{2}=[0;0;0];Pc{3}=[l1;0;0];Pc{4}=[l2;0;0];
%the mass of each link
m{2}=0;m{3}=m1;m{4}=m2;
I{2}=[0;0;0];I{3}=[0;0;0];
%the base of robot is rotating
w{1}=[0;0;q0];
dw{1}=[0;0;dq0];
%the rotating joints
dq{2}=[0;0;dq0];dq{3}=[0;dq1;0];dq{4}=[0;dq2;0];
ddq{2}=[0;0;ddq0];ddq{3}=[0;ddq1;0];ddq{4}=[0;ddq2;0];
%no force exerted on end-effector
f{5}=[0;0;0];
n{5}=[0;0;0];
%% Outward iteration
for i=1:3
w{i+1}=R{i}.'*w{i}+dq{i+1};
dw{i+1}=R{i}.'*dw{i}+cross(R{i}.'*w{i},dq{i+1})+ddq{i+1};
dv{i+1}=R{i}.'*(cross(dw{i},P{i})+cross(w{i},cross(w{i},P{i}))+dv{i});
dvc{i+1}=cross(dw{i+1},Pc{i+1})+cross(w{i+1},cross(w{i+1},Pc{i+1}))+dv{i+1};
F{i+1}=m{i+1}*dvc{i+1};
%Momement of inertia is zero
N{i+1}=[0;0;0];
end
%% Inward iteration
for i=4:-1:2
    f{i}=R{i}*f{i+1}+F{i};
    n{i}=N{i}+R{i}*n{i+1}+cross(Pc{i},F{i})+cross(P{i},R{i}*f{i+1});
end
%% torque
tau3=n{4}(3,1);
tau2=n{3}(3,1);
tau1=n{2}(3,1);

t1 = l2*m2*s2*s3*(l2*(s3*(s2*(ddq0 + dq0) + c2*dq1*(dq0 + q0)) - c3*(c2*(ddq0 + dq0) -...
    dq1*s2*(dq0 + q0)) + dq2*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0))) - c1*g - l1*(c2*(ddq0 + dq0) - ...
    dq1*s2*(dq0 + q0)) + l2*(dq1 + dq2)*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0)) + dq1*l1*s2*(dq0 + q0))
- ...
    c2*(l1*m2*(l2*(s3*(s2*(ddq0 + dq0) + c2*dq1*(dq0 + q0)) - c3*(c2*(ddq0 + dq0) - dq1*s2*(dq0 +
q0)) + ...
    dq2*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0))) - c1*g - l1*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) +
l2*(dq1 + ...
    dq2)*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0)) + dq1*l1*s2*(dq0 + q0)) - l1*m1*(c1*g + l1*(c2*(ddq0 +
dq0) -...
    dq1*s2*(dq0 + q0)) - dq1*l1*s2*(dq0 + q0)) + c3*l2*m2*(l2*(s3*(s2*(ddq0 + dq0) + c2*dq1*(dq0 +
q0)) - ...
    c3*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) + dq2*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0))) - c1*g -
l1*(c2*(ddq0 + dq0)...
    - dq1*s2*(dq0 + q0)) + l2*(dq1 + dq2)*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0)) + dq1*l1*s2*(dq0 +
q0)));
```

```
t2 = l1*m1*(c1*g + l1*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) - dq1*l1*s2*(dq0 + q0)) -
l1*m2*(l2*(s3*(s2*(ddq0 + dq0) +...
    c2*dq1*(dq0 + q0)) - c3*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) + dq2*(c2*s3*(dq0 + q0) +
c3*s2*(dq0 + q0))) - c1*g - ...
    l1*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) + l2*(dq1 + dq2)*(c2*s3*(dq0 + q0) + c3*s2*(dq0 + q0)) +
dq1*l1*s2*(dq0 + q0)) - ...
    c3*l2*m2*(l2*(s3*(s2*(ddq0 + dq0) + c2*dq1*(dq0 + q0)) - c3*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0))
+ dq2*(c2*s3*(dq0 + q0) + ...
    c3*s2*(dq0 + q0))) - c1*g - l1*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) + l2*(dq1 + dq2)*(c2*s3*(dq0 +
q0) + c3*s2*(dq0 + q0)) +...
    dq1*l1*s2*(dq0 + q0));

t3 = -l2*m2*(l2*(s3*(s2*(ddq0 + dq0) + c2*dq1*(dq0 + q0)) - c3*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0))
+ dq2*(c2*s3*(dq0 + q0) +...
    c3*s2*(dq0 + q0))) - c1*g - l1*(c2*(ddq0 + dq0) - dq1*s2*(dq0 + q0)) + l2*(dq1 + dq2)*(c2*s3*(dq0 +
q0) + c3*s2*(dq0 + q0)) +...
    dq1*l1*s2*(dq0 + q0));
```

4.2 Lagrange-Euler Formulation

Another method for simulating the dynamics of robotic arms is the Lagrange-Euler (L-E) formulation [10]. It is a mathematical method for describing a system's dynamics using Lagrange's equations of motion. The Lagrange-Euler formulation of a robotic arm entails applying the Lagrange equations to derive the system's equations of motion [12]. To determine the equations of motion, first calculate the system's kinetic and potential energies. Next, take the Lagrange's derivative with respect to the joint variables. Compared to the Newton-Euler formulation, the Lagrange-Euler formulation is more mathematically elegant and easier to use for complex systems with many degrees of freedom [13]. However, it can be more computationally expensive and requires more advanced mathematical knowledge to implement.

4.3 L-E: Computation for position, velocity and acceleration

Lagrange-Euler method requires the knowledge of linear velocity of each joint on robotic arm. As the robotic arm used in our assignment is 3 DOF rotational manipulator, one rotational base with respect to z-axis, two rotational joint with respect to y-axis, the position of joint one, two and three can be found geometrically:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{cases} 0 \\ 0 \end{cases} \qquad \text{`(Eq.32)}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{cases} l_1 c_1 + l_2 c_{12} \\ l_1 s_1 + l_2 s_{12} \end{cases} \tag{Eq.33}$$

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{cases} x_2 + l_3 c_{123} \\ y_2 + l_3 s_{123} \end{cases} \tag{Eq.34}$$

where $s_1$means sin of first joint angle, s_{12} means sin of first joint angle plus second joint angle, etc.

From position of each joint, the velocity of each joint in x and y direction can be found by taking derivative calculation with respect to time of the above expression:

$$\begin{bmatrix} \dot{x_1} \\ \dot{y_1} \end{bmatrix}_L = \begin{cases} 0 \\ 0 \end{cases} \tag{Eq.35}$$

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix}_L = \begin{cases} -(l_1 s_1 \dot{q}_1 + l_2 c_{12}(\dot{q}_1 + \dot{q}_2)) \\ \ l_1 c_1 \dot{q}_1 + l_2 s_{12}(\dot{q}_1 + \dot{q}_2) \end{cases} \tag{Eq.36}$$

$$\begin{bmatrix} \dot{x_3} \\ \dot{y_3} \end{bmatrix}_L = \begin{cases} -l_1 s_1 \dot{q}_1 - l_2 s_{12}(\dot{q}_1 + \dot{q}_2) - l_3 s_{123}(\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\ \ l_1 c_1 \dot{q}_1 + l_2 c_{12}(\dot{q}_1 + \dot{q}_2) + l_3 c_{123}(\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \end{cases} \tag{Eq.37}$$

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix}_\omega = \omega_1 \tag{Eq.38}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{y_1} \end{bmatrix}_\omega = \omega_1 \tag{Eq.39}$$

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix}_\omega = \omega_1 \tag{Eq.40}$$

Therefore, the square of velocity vector of each joint now can be established as:

$$v_i{}^2 = x_i{}^2 + y_i{}^2 \tag{Eq.41}$$

Now, with linear velocity representation, the Lagrange-Euler equation can be interpreted as:

$$\boldsymbol{\frac{d}{dt}} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau, \ \ L = K - P \tag{Eq.42}$$

where K and P are kinetic energy and potential energy of the robotic arm.

First of all, the kinetic energy can be found as below, where m and v are the mass and velocity of the link and joint:

$$K_i = \frac{1}{2}m_i v_i^2 + \frac{1}{2}I_i\omega_i^2 \qquad \text{(Eq.43)}$$

Next, the potential energy can be derived, where m and h are mass and height of the link and joint:

$$P_i = m_i g h_i \qquad \text{(Eq.44)}$$

Key codes

```
clear;close all;clc;
%% L-E dynamic formulation - 3DOF RRR manipulator
syms M1 M2 M3;
syms x1 x1d x1dd x2 x2d x2dd x3 x3d x3dd I1 I2 I3 omega;
syms L1 L2 L3;
syms u1 u2 u3;
syms g

% Position
p1x = 0;
p1y = 0;
p2x = p1x+L2*cos(q1+q2);
p2y = p1y+L2*sin(q1+q2);
p3x = p2x+L3*cos(q1+q2+q3);
p3y = p2y+L3*sin(q1+q2+q3);

% Velocity
v1x = 0;
v1y = 0;
v2x = v1x-L2*sin(q1)*(q1d);
v2y = v1y+L2*cos(q1)*(q1d);
v3x = v2x - L3*sin(q1+q2)*(q1d+q2d);
v3y = v2y + L3*cos(q1+q2)*(q1d+q2d);

% Kinetic energy
KE = 0.5*M1*( v1x^2 + v1y^2)+0.5*I1 *omega^2 + 0.5*M2*( v2x^2 + v2y^2)+0.5*I2*omega^2 +
0.5*M3*( v3x^2 + v3y^2)+0.5*omega^2*I3;
KE = simplify(KE);

% Potential energy
PE = M1*g*p1y + M2*g*p2y + M3*g*p3y;
PE = simplify(PE);

Px1 = u1;
Px2 = u2;
Px3 = u3;
% dk/dx1d
pKEpx1d = diff(KE,q1d);
% d/dt (dk/dx1d)
ddtpKEpx1d = diff(pKEpx1d,q1)*q1d+ ...
         diff(pKEpx1d,q1d)*q1dd+ ...
```

```
            diff(pKEpx1d,q2)*q2d + ...
            diff(pKEpx1d,q2d)*q2dd + ...
            diff(pKEpx1d,q3)*q3d + ...
            diff(pKEpx1d,q3d)*q3dd;
% KE derivative wrt q1
pKEpx1 = diff(KE,q1);
% P derivative wrt q1
pPEpx1 = diff(PE,q1);
% PE derivative wrt q2 dot, q2
pKEpx2d = diff(KE,q2d);
ddtpKEpx2d = diff(pKEpx2d,q1)*q1d+ ...
            diff(pKEpx2d,q1d)*q1dd+ ...
            diff(pKEpx2d,q2)*q2d + ...
            diff(pKEpx2d,q2d)*q2dd + ...
            diff(pKEpx2d,q3)*q3d + ...
            diff(pKEpx2d,q3d)*q3dd;
pKEpx2 = diff(KE,q2);
pPEpx2 = diff(PE,q2);

%PE derivative wrt q3 dot, q3
pKEpx3d = diff(KE,q3d);
ddtpKEpx3d = diff(pKEpx3d,q1)*q1d+ ...
            diff(pKEpx3d,q1d)*q1dd+ ...
            diff(pKEpx3d,q2)*q2d + ...
            diff(pKEpx3d,q2d)*q2dd + ...
            diff(pKEpx3d,q3)*q3d + ...
            diff(pKEpx3d,q3d)*q3dd;
pKEpx3 = diff(KE,q3);
pPEpx3 = diff(PE,q3);


out_q1 = simplify( ddtpKEpx1d - pKEpx1 + pPEpx1 - Px1);
out_q2 = simplify( ddtpKEpx2d - pKEpx2 + pPEpx2 - Px2);
out_q3 = simplify( ddtpKEpx3d - pKEpx3 + pPEpx3 - Px3);
```

Thanks to the inbuilt ‘diff’ function in MATLAB, the differential function can be much easier to solve. The three output torque equations are shown below:

Be taking partial differentiation with respect to $\ddot{q}, \dot{q}$, the Lagrange equation can be transferred into matrix form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad \text{(Eq.45)}$$

where the mass inertia matrix, M, Ceriolis and Centrifugal matrix C are derived from MATLAB code:

```
M(1,1) = (1/3).*L1.^2.*m1+L1.^2.*m2+(1/3).*L2.^2.*m2+L1.^2.*m3+L2.^2.*m3+(1/3).*...
  L3.^2.*m3+(1/12).*m1.*W.^2+(1/12).*m2.*W.^2+(1/12).*m3.*W.^2+L1.*L2.*...
  m2.*cos(q2)+2.*L1.*L2.*m3.*cos(q2)+L2.*L3.*m3.*cos(q3)+L1.*...
  L3.*m3.*cos(q2+q3);
M(1,2) = (1/3).*L2.^2.*m2+L2.^2.*m3+(1/3).*L3.^2.*m3+(...
  1/12).*m2.*W.^2+(1/12).*m3.*W.^2+(1/2).*L1.*L2.*m2.*cos(q2)+L1.*L2.*...
  m3.*cos(q2)+L2.*L3.*m3.*cos(q3)+(1/2).*L1.*L3.*m3.*cos(q2+...
  q3);
M(1,3) = (1/3).*L3.^2.*m3+(1/12).*m3.*W.^2+(1/2).*L2.*L3.*m3.*cos(q3)+...
  (1/2).*L1.*L3.*m3.*cos(q2+q3);
M(2,1) = M(1,2);
M(2,2) = (1/3).*L2.^2.*m2+L2.^2.*m3+(1/3).*L3.^2.*m3+(...
  1/12).*m2.*W.^2+(1/12).*m3.*W.^2+L2.*L3.*m3.*cos(q3);
M(2,3) =(1/3).*L3.^2.*...
  m3+(1/12).*m3.*W.^2+(1/2).*L2.*L3.*m3.*cos(q3) ;
M(3,1) = M(1,3);
M(3,2) = M(2,3);
M(3,3) = (1/3).*L3.^2.*m3+(1/12).*m3.*W.^2;

 C = zeros(3,3);
 C1(1,1) = (-1).*dq2.*L1.*L2.*m2.*sin(q2)+(-2).*dq2.*L1.*L2.*m3.* ...
   sin(q2)+(-1).*dq3.*L2.*L3.*m3.*sin(q3)+(-1).*dq2.* ...
   L1.*L3.*m3.*sin(q2+q3)+(-1).*dq3.*L1.*L3.*m3.*sin(q2+ ...
   q3);
 C1(1,2) = (-1).*dq1.*L1.*L2.*m2.*sin(q2)+(-1).*dq2.*L1.* ...
   L2.*m2.*sin(q2)+(-2).*dq1.*L1.*L2.*m3.*sin(q2)+(-2).* ...
   dq2.*L1.*L2.*m3.*sin(q2)+(-1).*dq3.*L2.*L3.*m3.*sin( ...
   q3)+(-1).*dq1.*L1.*L3.*m3.*sin(q2+q3)+(-1).* ...
   q2.*L1.*L3.*m3.*sin(q2+q3)+(-1).*dq3.*L1.*L3.*m3.* ...
   sin(q2+q3);
 C1(1,3) = (-1).*dq1.*L2.*L3.*m3.*sin(q3)+(-1).* ...
   dq2.*L2.*L3.*m3.*sin(q3)+(-1).*dq3.*L2.*L3.*m3.*sin( ...
   q3)+(-1).*dq1.*L1.*L3.*m3.*sin(q2+q3)+(-1).* ...
   dq2.*L1.*L3.*m3.*sin(q2+q3)+(-1).*dq3.*L1.*L3.*m3.* ...
   sin(q2+q3);
 C1(2,1) = (-1/2).*dq2.*L1.*L2.*m2.*sin(q2)+(-1).* ...
   dq2.*L1.*L2.*m3.*sin(q2)+(-1).*dq3.*L2.*L3.*m3.*sin( ...
   q3)+(-1/2).*dq2.*L1.*L3.*m3.*sin(q2+q3)+(-1/2).* ...
   dq3.*L1.*L3.*m3.*sin(q2+q3);
 C1(2,2) = (-1/2).*dq1.*L1.*L2.* ...
   m2.*sin(q2)+(-1).*dq1.*L1.*L2.*m3.*sin(q2)+(-1).* ...
   dq3.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin( ...
   q2+q3);

C1(2,3) =(-1).*dq1.*L2.*L3.*m3.*sin(q3)+(-1).* ...
  dq2.*L2.*L3.*m3.*sin(q3)+(-1).*dq3.*L2.*L3.*m3.*sin( ...
  q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin(q2+q3);
C1(3,1) = (-1/2).* ...
  dq3.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq2.*L1.*L3.*m3.*sin( ...
  q2+q3)+(-1/2).*dq3.*L1.*L3.*m3.*sin(q2+q3);
C1(3,2) =(-1/2).* ...
  dq3.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin( ...
  q2+q3);
C1(3,3) =(-1/2).*dq1.*L2.*L3.*m3.*sin(q3)+(-1/2).* ...
  dq2.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin( ...
  q2+q3);
```

```
C2(1,1) = 0;
C2(1,2) = 0;
C2(1,3) = 0;
C2(2,1) =(-1).*dq1.*L1.*L2.*m2.*sin(q2)+(-1/2).*dq2.*L1.* ...
  L2.*m2.*sin(q2)+(-2).*dq1.*L1.*L2.*m3.*sin(q2)+(-1).* ...
  dq2.*L1.*L2.*m3.*sin(q2)+(-1).*dq1.*L1.*L3.*m3.*sin( ...
  q2+q3)+(-1/2).*dq2.*L1.*L3.*m3.*sin(q2+q3)+(-1/2).* ...
  dq3.*L1.*L3.*m3.*sin(q2+q3);
C2(2,2) =(-1/2).*dq1.*L1.*L2.* ...
  m2.*sin(q2)+(-1).*dq1.*L1.*L2.*m3.*sin(q2)+(-1/2).* ...
  dq1.*L1.*L3.*m3.*sin(q2+q3);
C2(2,3) = (-1/2).*dq1.*L1.*L3.* ...
  m3.*sin(q2+q3);
C2(3,1) =(-1).*dq1.*L2.*L3.*m3.*sin(q3)+(-1).* ...
  dq2.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq3.*L2.*L3.*m3.*sin( ...
  q3)+(-1).*dq1.*L1.*L3.*m3.*sin(q2+q3)+(-1/2).* ...
  dq2.*L1.*L3.*m3.*sin(q2+q3)+(-1/2).*dq3.*L1.*L3.* ...
  m3.*sin(q2+q3);
C2(3,2) =(-1).*dq1.*L2.*L3.*m3.*sin(q3)+(-1).* ...
  dq2.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq3.*L2.*L3.*m3.*sin( ...
  q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin(q2+q3);
C2(3,3) =(-1/2).* ...
  dq1.*L2.*L3.*m3.*sin(q3)+(-1/2).*dq2.*L2.*L3.*m3.*sin( ...
  q3)+(-1/2).*dq1.*L1.*L3.*m3.*sin(q2+q3);
```

Lastly, gravity term matrix, G is shown below:

```
G(1) = (1/2).*g0.*(L1.*(m1+2.*(m2+m3)).*cos(q1)+L2.*(m2+2.*m3).*cos(q1+ ...
  q2)+L3.*m3.*cos(q1+q2+q3));
G(2) = (1/2).*g0.*(L2.*(m2+2.*m3).* ...
  cos(q1+q2)+L3.*m3.*cos(q1+q2+q3));
G(3) = (1/2).*g0.*L3.*m3.* ...
  cos(q1+q2+q3);
```

4.4 Comparation between N-E and L-E

By using equations calculated above we give the same parameters and initial state. The torque energy - time curve of two methods are very similar. As shown in figure XX, the torque fluctuation of the Newton-Euler curve is smaller than the Lagrange-Euler curve. Variation range of joint angle is approximately 0.7rad. The biggest angular speed is around 0.1rad/s. The result also approve that the final result of N-E and L-E are almost the same even though the calculation processing is different.

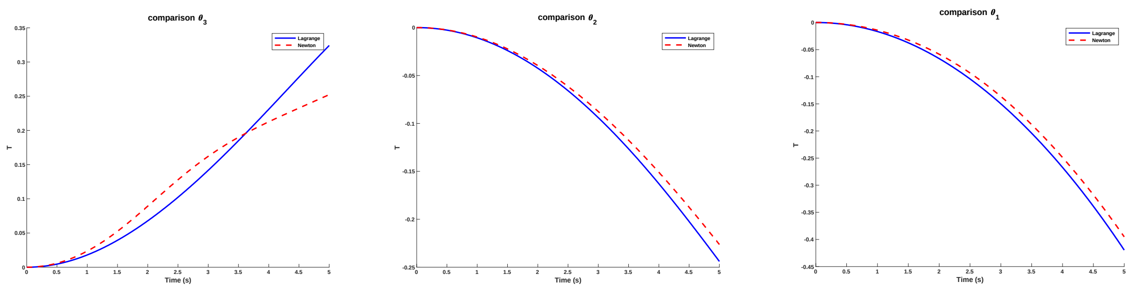


Figure 10 Comparation between N-E and L-E

Chapter 5 Trajectory tracking based on PID controller

5.1 PID control design

PID is a widely-used control algorithm in industrial and automation control. It stands for proportional-integral-derivative controller, and it calculates the controller's output signal based on the error between the feedback signal from the controlled object and the desired input signal. The three components of PID (proportional, integral, and differential part) work together to control the motion of the controlled object.

The proportional component generates the controller's output signal based on the error signal magnitude. The integral component generates the output signal based on the error signal's integral value over time. The differential component generates the output signal based on the error signal's differential value over time. These three components produce output signals that represent the controller's instantaneous response, steady-state response, and transition response.

The whole control system for trajectory tracking is shown below.

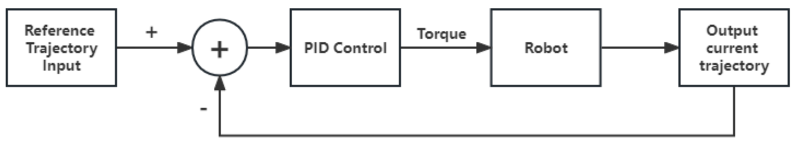


Figure 11 PID control system structure

And PID control part is given by this equation:

$$\tau = K_P * error + K_I \int error + K_D * \dot{error} \qquad \text{(Eq.46)}$$

For simulating RRR manipulator PID control in Simulink, first thing is to build robot model in Simulink.

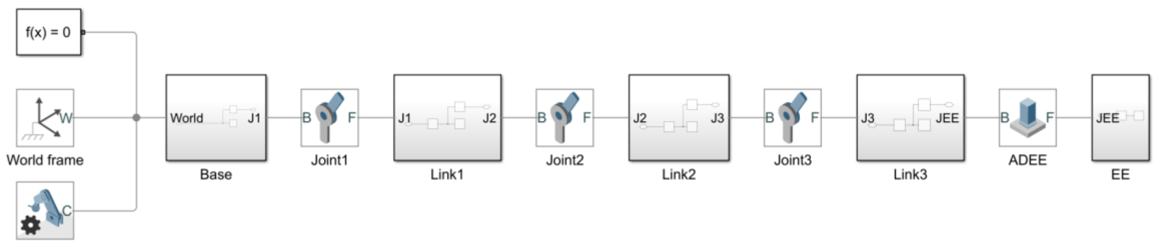


Figure 12 Robot rigid body tree

Here, frame block is used for set reference fame for each joint and link. For each detailed Link block, frame should be set properly.

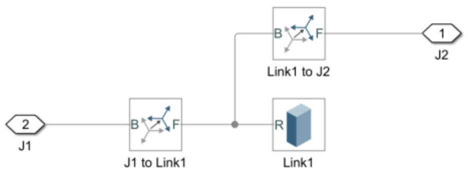


Figure 13 Each joint structure

The graph shows the simulating of RRR manipulator obtained by the model.

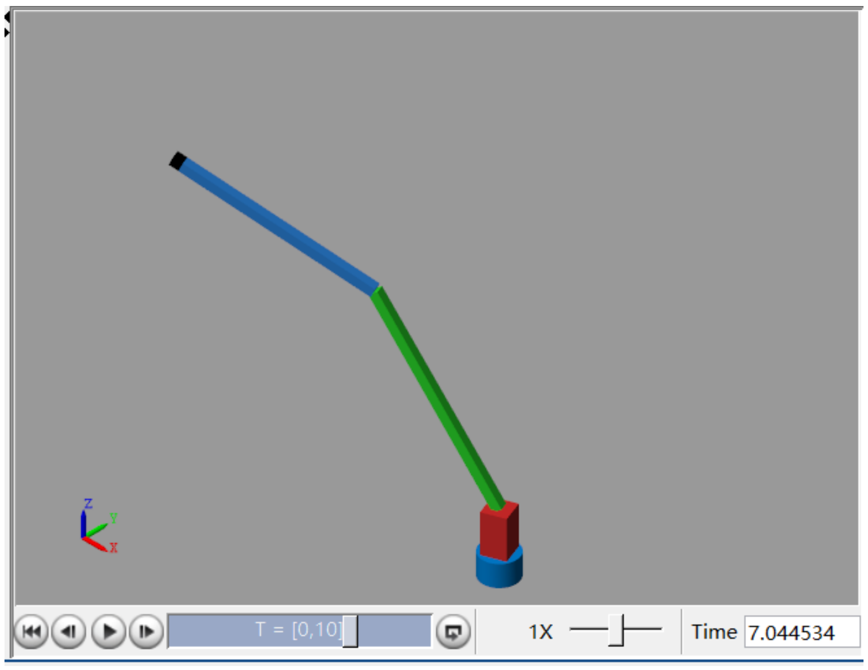


Figure 14 Build RRR manipulator

Then, PID blocks are added to the system. The inputs of PID blocks are the desired trajectory value (theta of each joint), and currently joint position, and output torque to robot. Three channel signal generator is used to set three joint trajectories. Three scopes are placed in the right areas named q1, q2, and q3 which separately shows the desired and physical trajectory of three joints.

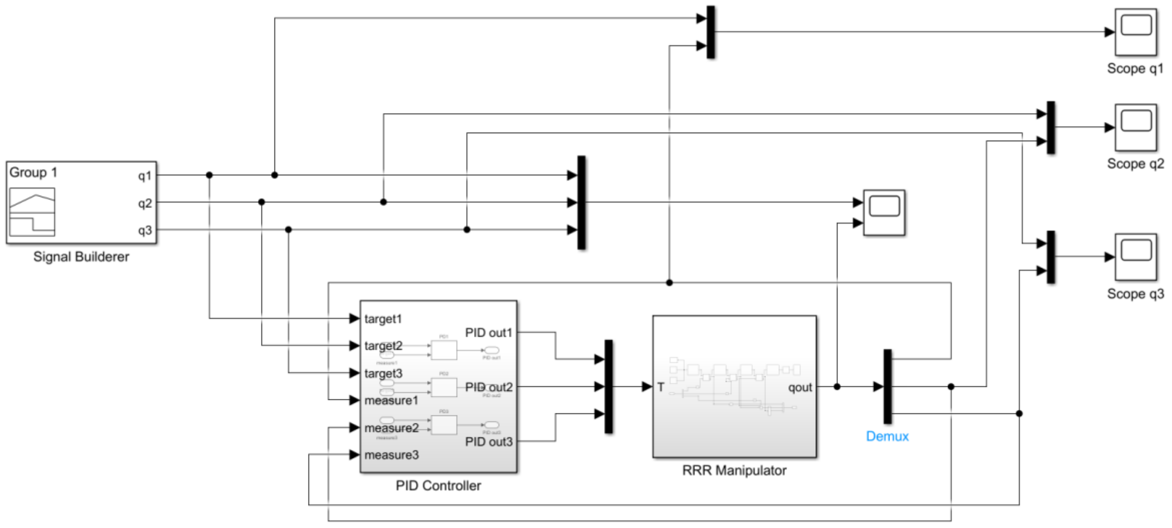
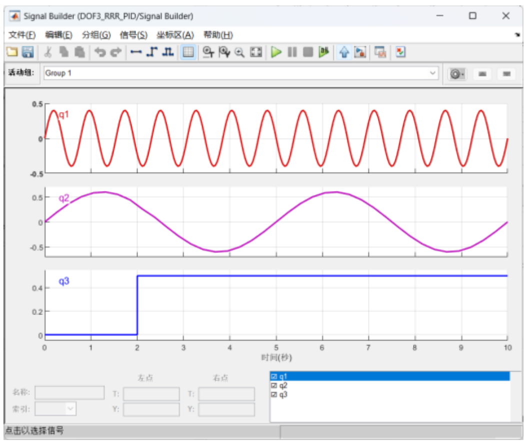


Figure 15 Manipulator with PID controller

5.2 PID result visualization

To test the PID controller, the signal builder block need to modified first. There are three signals, as the input for each joint theta value, including sine function and step function.



Signal Builder (DOF3_RRR_PID/Signal Builder)
文件(F) 编辑(E) 分组(G) 信号(S) 坐标区(A) 帮助(H)
活动组: Group 1
q1
q2
q3
时间(秒)
点击以选择信号

Figure 16 Signals generator

Without setting PID values, a default value for each parameter is chosen. The result is shown below, the upper window shows the desired trajectory and the lower window shows the trajectory of the physical movement of the robot arm through PID control.

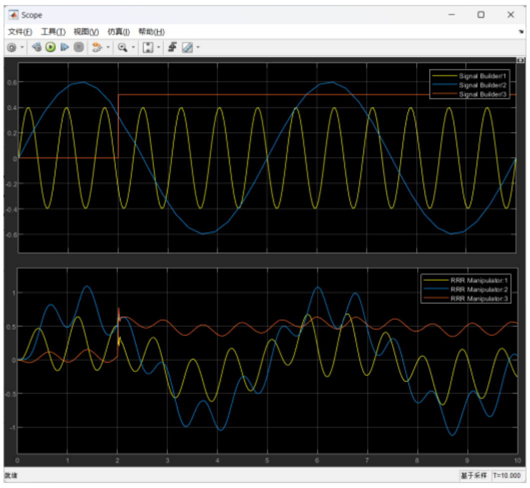


Figure 17 Compared for desired and physical trajectory

To visualize clearly, three separate joint comparisons are given. The yellow line represents target trajectory while the blue one is physical trajectory under PID controller.

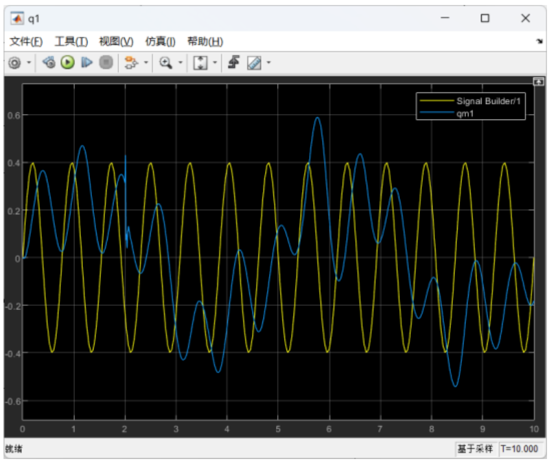


Figure 18 Compared for desired and physical trajectory for joint 1

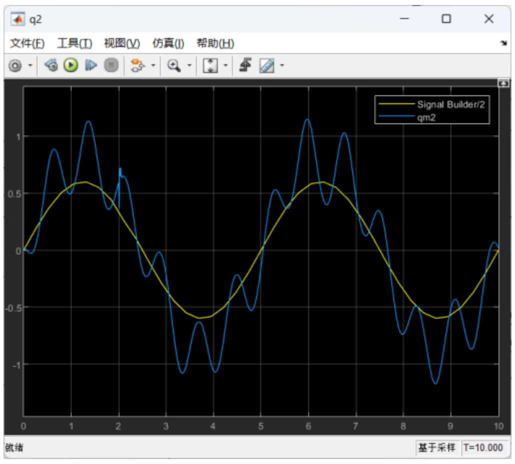


Figure 19 Compared for desired and physical trajectory for joint 2

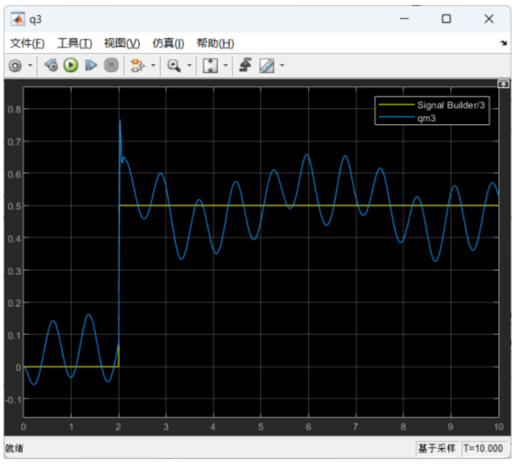


Figure 20 Compared for desired and physical trajectory for joint 3

These results are obtained without optimisation of the PID parameters, which means, for PID controller, the most important thing is setting appropriate parameters to get a stable response. This diagram shows the better control results that can be achieved with a suitable set PID values for joint1.

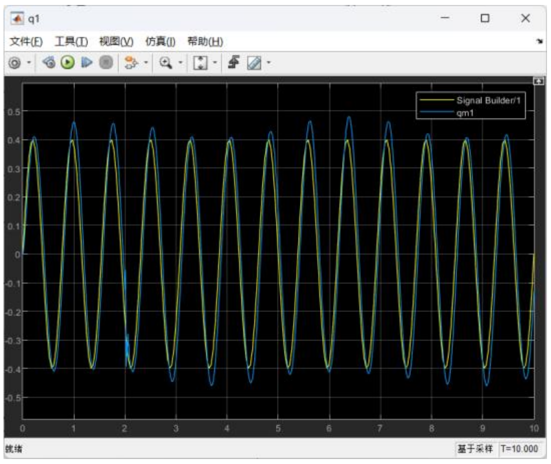


Figure 21 Compared for desired and physical trajectory for joint 1

Refer to figure 23, it demonstrates a better performance compared with figure 20. The performance of a PID controller is highly dependent on the setting of the three PID parameters, so the biggest challenge for a PID controller is to determine a suitable parameter value to ensure that the output will respond well

5.3 How each PID parameter influence robot performance

To illustrate how each PID parameter influence the performance, the input of joint 3 is set to a step signal, which can show the effect for different PID value easily.

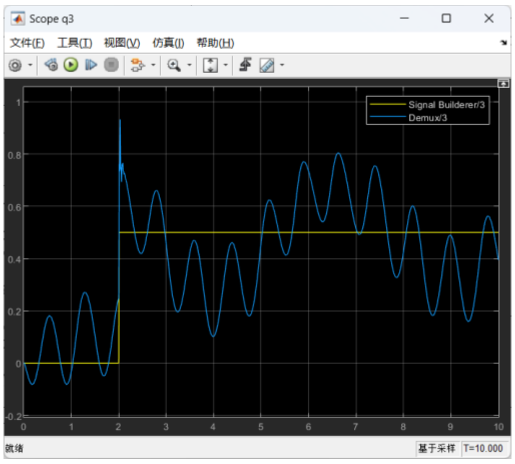


Figure 22 Performance under P=1,I=1,D=1

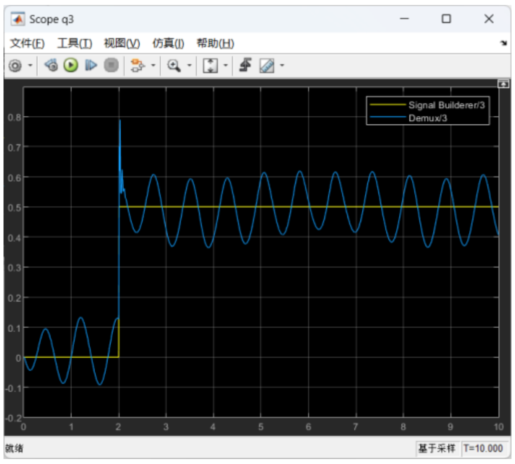


Figure 23 Performance under P=15,I=1,D=1

From figure 24 and figure 25, it clearly demonstrates how changing the P-value affects the output performance. With larger P-value, faster response of the PID controller will be, making the controller more sensitive to error signals. However, too large P-value can easily lead to oscillations and overshooting of the controller output signal, which can affect the stability of the control effect.

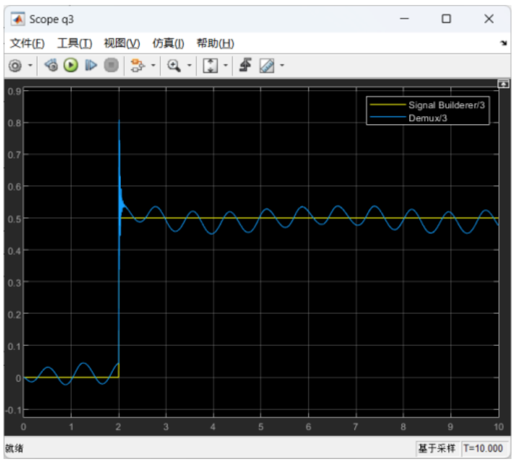


Figure 24 Performance under P=15,I=1,D=5

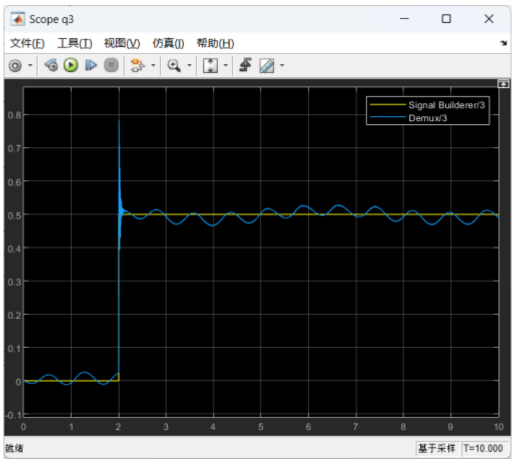


Figure 25 Performance under P=15,I=1,D=10

From figure 26 and figure 27, when the parameter D is increased, the transition response of the PID controller is enhanced. The controller is more sensitive to the rate of change of the error signal. As a result, the response of the controller is faster and the controller output signal is adjusted more quickly to near the desired value, thus reducing the amount of overshoot and oscillation. This helps to improve the stability and accuracy of the control system.

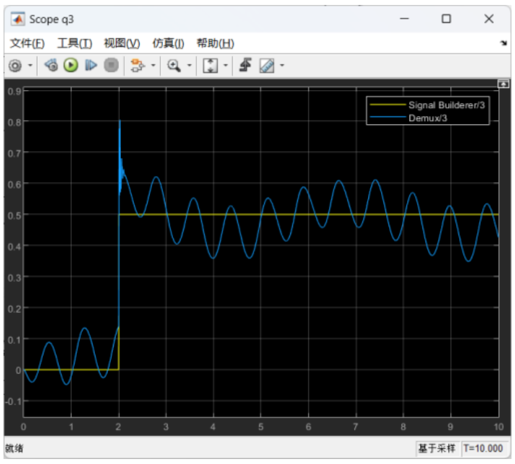


Figure 26 Performance under P=3,I=0,D=2

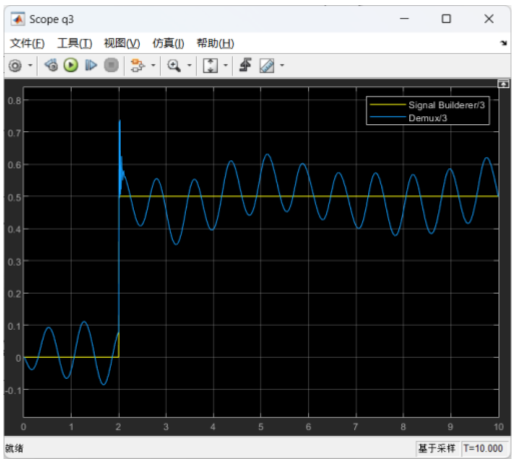


Figure 27 Performance under P=3,I=10,D=2

From figure 28 and figure 29, it shows that when the parameter I is increased, the steady-state response of the PID controller is enhanced, thus reducing the steady-state error of the system and improving the accuracy of the control system. However, it should be noted that too large I value may lead to over-regulation of the controller, resulting in over-compensation of the system and thus affecting the stability of the control effect.

Chapter 6 Trajectory tracking based on computed torque control

6.1 Computed torque controller design

The Computed Torque Method is a dynamical model-based control method that allows for highly accurate and robust robot control. This method achieves accurate control of the robot by modelling the robot dynamics and solving for the control torque using computational methods. The method first calculates the control torque based on the robot's dynamics model, then uses the position, velocity and acceleration of the reference model as the desired input signals, calculates the desired torque and finally performs feedback control to regulate the robot's motion based on the error signal.

In general, the dynamics of a robot can be modelled by the following equation:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad \text{(Eq.47)}$$

where M(q) is the joint mass matrix, q, $\dot{q}$, $\ddot{q}$ are the joint position, velocity and acceleration respectively, C (q, q') is the Coriolis force matrix, G(q) is the gravity matrix and τ is the joint moment of the robot.

Therefore, the computed torque control can be obtained:

$$\tau = M(q)\left[\ddot{q}_r + K_d(\dot{q}_r - \dot{q}) + K_p(q_r - q)\right] + C(q,\dot{q})(\dot{q}_r - \dot{q}) + G(q) \qquad \text{(Eq.48)}$$

According to the equation, a CTC computing model could be established in Simulink as below:

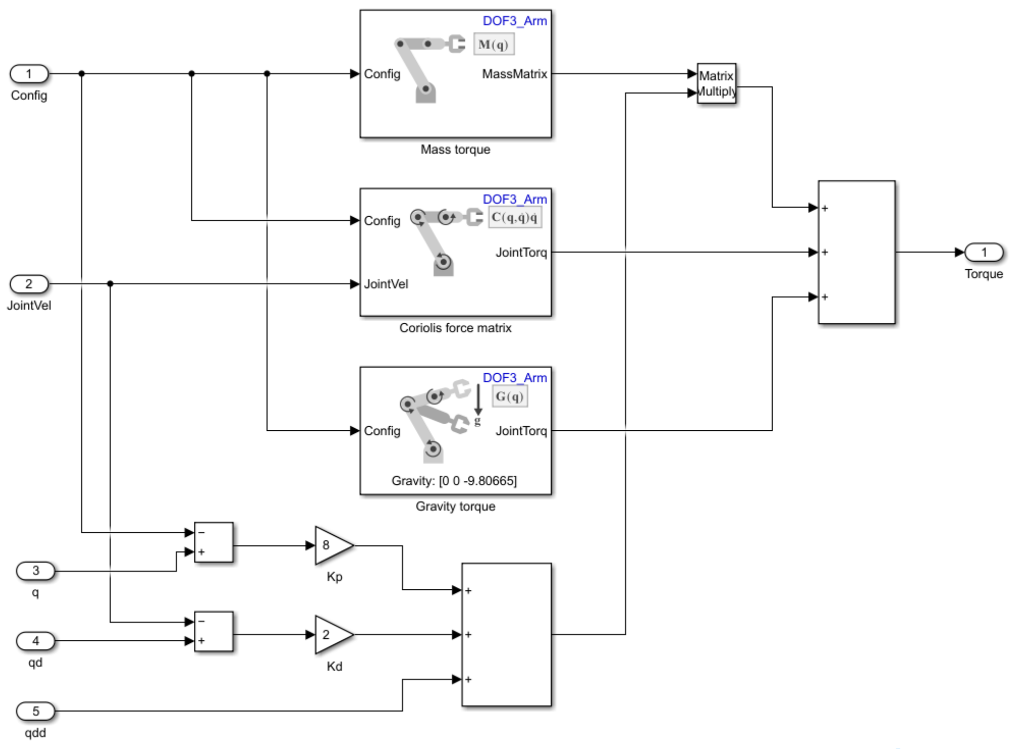


Figure 28 CTC computing block

DOF3_Arm in the file is the rigid body tree (refer to figure 15) of the RRR manipulator. The whole CTC controller is built in Figure 30.

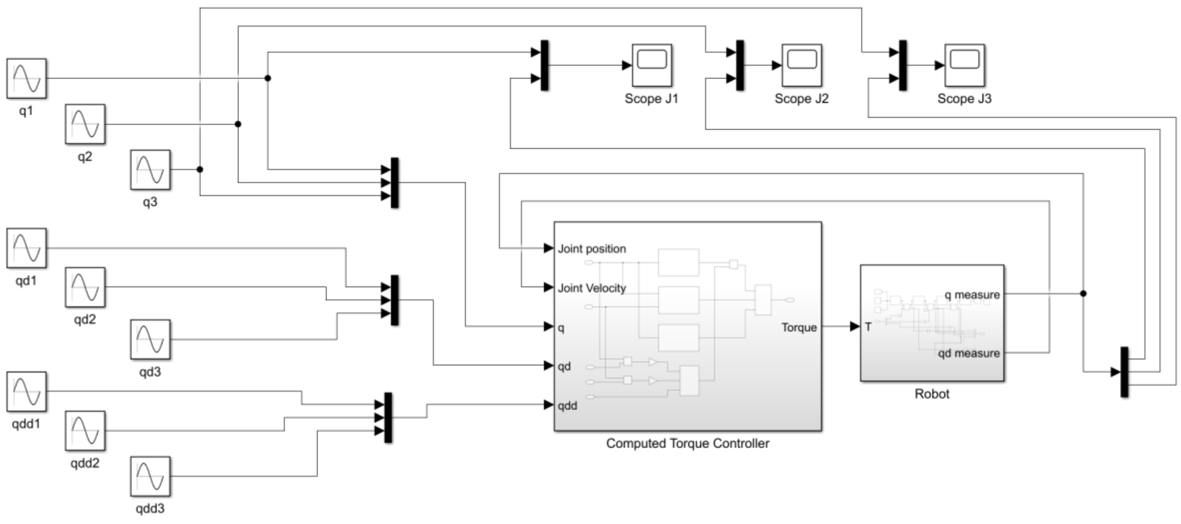


Figure 29 CTC Controller

Refer to figure 31, it shows the structure of CTC controller. The reference trajectory is generated by the sine wave generator on the left. where qd1 is the first-order derivative of the q1 signal and qdd1 is the second-order derivative of the q1 signal, the same as q2 and q3. After computing the torque needed for robot, there are three scopes for visualize each joint target trajectory and physical trajectory.

6.2 CTC result visualization

To simulate the CTC controller, first step is providing reference trajectory. Figure 32 depicts the input trajectory for joint 1 with its first-order and second-order derivatives.

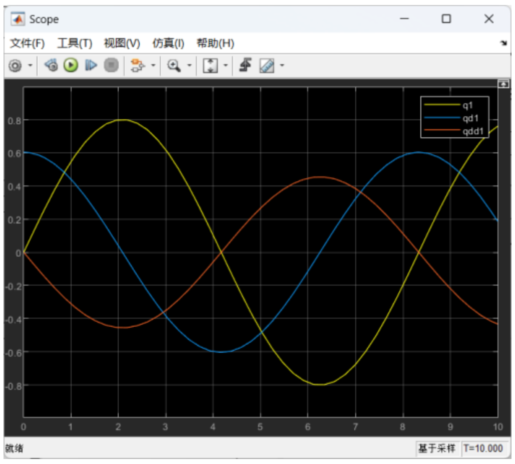


Figure 30 Reference trajectory with derivatives for joint 1

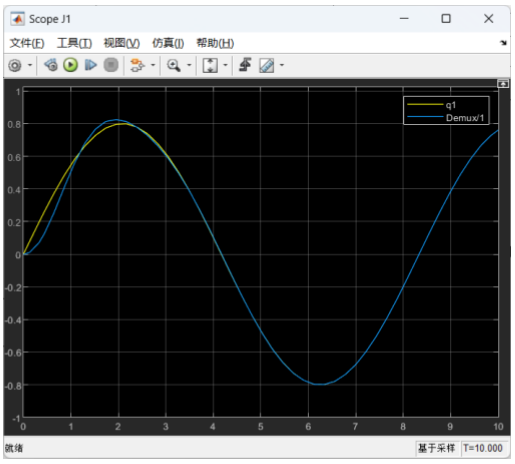


Figure 31 Compared target and physical trajectory for joint 1

The yellow line is targeting trajectory, the blue one is physical trajectory under CTC control method, which almost looks the same. This also identify the strong trajectory tracking capability of the CTC control method, as well as the high accuracy.

6.3 Compare with PID control

For PID controllers, the output needs to reach a target value, but the target trajectory point may have changed before the output of PID controller is reached. In this case, the error is always present and varies, which means that PID controller is not well suited for trajectory tracking task. computed torque controller, The CTC controller allows for more accurate trajectory tracking control. This is because CTC controllers are based on the robot dynamics model and can better take into account the dynamic characteristics of the robot. But PID controllers have better immunity to interference and can maintain better trajectory tracking in the presence of noise, friction and other disturbing factors. This is because the PID controller is based on the error signal for control, can be achieved through feedback control to suppress and compensate for disturbing signals. Also, for parameters setting, it is easier to optimize PID control as there are only three parameters. If optimize CTC controller, there are a lot of parameters needed to adjust which is a complex processing.

Chapter 7 Conclusion and future work

7.1 Summarize and improvement

In this report we propose an RRR Robot and derive its kinematics as well as his dynamics. In the kinematics section we use MATLAB to calculate forward kinematics, inverse kinematics, and Jacobian matrices. In the dynamics section, we derived the n-Newton Eulerian and Lagrange Eulerian equations for the robot. and implemented for a given robot input torque. The angles, angular velocities and angular accelerations of each of its joints are calculated. In addition, two controllers have been designed for the trajectory tracking task, the PID controller and the CTC controller. And the influence of each PID parameter d on the results is discussed in detail. The two controllers are also modelled in Simulink and the results of the CTC controller are compared with those of the PID controller. The results show that the CTC controller can perform the trajectory tracking task perfectly.

Actually, our model is very simple and basic which means there is a lot of space to improve. As we mentioned above, It exists the singularity problem when we use Jacobian method to control the speed of robot. So, we need consider singularity analysis to improve the performance if implement it. As for the control method, there are many advanced control algorisms such as Neural Network Control (we already learnt in class), Model Predictive Control (MPC) and so on. Furthermore, the robot model can also be improved by introducing more joints and more advanced models. So, we have a long way to go.

7.2 Other technologies could be combined

The scenario of surgery robot and computer-assisted surgery robot is an interdisciplinary subject, many technologies can be integrated. For example, haptic feedback and augmented reality are two very helpful technology can be combined.

Haptic feedback technology allows the robot to simulate the sense of touch, providing the surgeon with a sense of physical feedback during the operation. This can enhance the precision and accuracy of the robot during surgery. Article [8] discusses the use of haptic feedback and forward/inverse kinematics in controlling twin serial robot manipulators installed on a modified UR5 robot. Field tests confirmed accurate haptic feedback and the sufficiency of current scaling coefficients. There are many methods to achieve this such as Tactile feedback, Electro vibration, Ultrasonic feedback and so on.

Augmented reality technology can also be integrated into the surgical robot, allowing the surgeon to see a 3D image of the patient's internal anatomy in real time. This can help the surgeon to better navigate the robot and accurately perform the surgery, reducing the risk of errors and complications. It can also provide additional information to the surgeon, such as blood flow and tissue thickness, that may not be visible to the naked eye. In paper [9] it describes an augmented reality-based training system for vascular interventional surgeons that uses gesture interaction instead of traditional interaction methods. Experimental results show that the system can improve the doctors' operational needs and training effectiveness.

Besides, there are also some other technologies. Artificial intelligence (AI) technology, which can be used to analyse large amounts of patient data and provide insights to the surgeon during the surgery, is a very hot topic in nowadays. Or computer version and so on.

Reference

[1] G. Kim, J. Park, T. Choi, H. Do, D. Park and J. Kyung, "Case studies of a industrial dual-arm robot application," *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, Korea (South), 2017, pp. 301-302, doi: 10.1109/URAI.2017.7992735.

[2] W. -T. Weng, H. -P. Huang and Y. -L. Zhao, "The Application of Dual-arm Mobile Robot Systems in the Care of the Older Adults," *2022 International Conference on System Science and Engineering (ICSSE)*, Taichung, Taiwan, 2022, pp. 127-132, doi: 10.1109/ICSSE55923.2022.9948250.

[3] H. R. Chowdhury, S. Hossain and M. H. Imam, "Design of Industrial Robotic Arm For Surgical Applications," *2023 3rd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, Dhaka, Bangladesh, 2023, pp. 331-335, doi: 10.1109/ICREST57604.2023.10070037.

[4] M. A. El Hamied, "Controlling Robotic Arm Assisted in Knee Surgery Utilizing Artificial Immunity Technique," *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, Cambridge, UK, 2013, pp. 266-270, doi: 10.1109/UKSim.2013.144.

[5] Universal Robot User Manual
https://www.usna.edu/Users/weaprcon/kutzer/_files/documents/User%20Manual,%20UR5.pdf

[6] A. R. Al Tahtawi, M. Agni, and T. D. Hendrawati, “Small-scale Robot Arm Design with Pick and Place Mission Based on Inverse Kinematics,” Journal of Robotics and Control (JRC), vol. 2, no. 6, pp. 469-475, 2021.

[7] M. Beedel, P. Zarafshan and A. A. Nazari, "Jacobian-based singularity analysis of a 3-CRRR spatial parallel robot," *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, Tehran, Iran, 2016, pp. 489-494, doi: 10.1109/ICRoM.2016.7886790.

[8] A. Lukin, G. L. Demidova, A. Rassõlkin, T. Vaimann and H. Roozbahani, "Force-based Feedback for Haptic Device of Mobile Assembly Robot," 2020 27th International Workshop on Electric Drives: MPEI Department of Electric Drives 90th Anniversary (IWED), Moscow, Russia, 2020, pp. 1-5, doi: 10.1109/IWED48848.2020.9069581.

[9] J. Guo, S. Jia and S. Guo, "A Novel Surgeon Training System for the Vascular Interventional Surgery based on Augmented Reality," *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, Takamatsu, Japan, 2021, pp. 1425-1430, doi: 10.1109/ICMA52036.2021.9512663.

[10] Shala, A., & Bajrami, X. (2013, May). Dynamic Modeling and Analysis of Propulsion effect of 3 DoF robot. *International Journal of Business & Technology*, *1*(2), 30–38. https://doi.org/10.33107/ijbte.2013.1.2.03

[11] Ata, A. A., Ghazy, M. A., & Gadou, M. A. (2013). Dynamics of a General Multi-axis Robot with Analytical Optimal Torque Analysis. *Journal of Automation and Control Engineering*, *1*(2), 144–148. https://doi.org/10.12720/joace.1.2.144-148

[12] Jangid, M. K., Kumar, S., & Singh, J. (2021, October 31). Trajectory tracking optimization and control of a three link robotic manipulator for application in casting. *International Journal of Advanced Technology and Engineering Exploration*, *8*(83). https://doi.org/10.19101/ijatee.2021.874468

[13] Dong, S., Yuan, Z., & Zhang, F. (2019, November 1). A Simplified Method for Dynamic Equation of Robot in Generalized Coordinate System. *Journal of Physics: Conference Series*, *1345*(4), 042077. https://doi.org/10.1088/1742-6596/1345/4/042077