ECSE 4480
ROBOTICS I

MiniProject 2: Planar Robot Arm Motion

I, Chuizheng Kong, certify that the following work is my own and completed in accordance with the academic integrity policy as described in the Robotics I course syllabus.

September 20, 2021

Chuizheng Kong

1 SUMMARY

In this project, I looked at simulations of planner linked robot arms with a pre-defined "S" shaped path. The goal of the project was to trace out the path with the end effector arm normal to the path. In the first part, the derivation of the forward and inverse kinematics were discussed. The method was then implemented in MATLAB to solve for the motion of a three-link robot.

2 TECHNICAL CONTENT

I first started with a three-linked arm robot with link lengths $(1.5,\ 1.5,\ 0.5)m$ rooted at point $(0,\ 0,\ 0)$. Following is a representation of the robot in zero configuration:

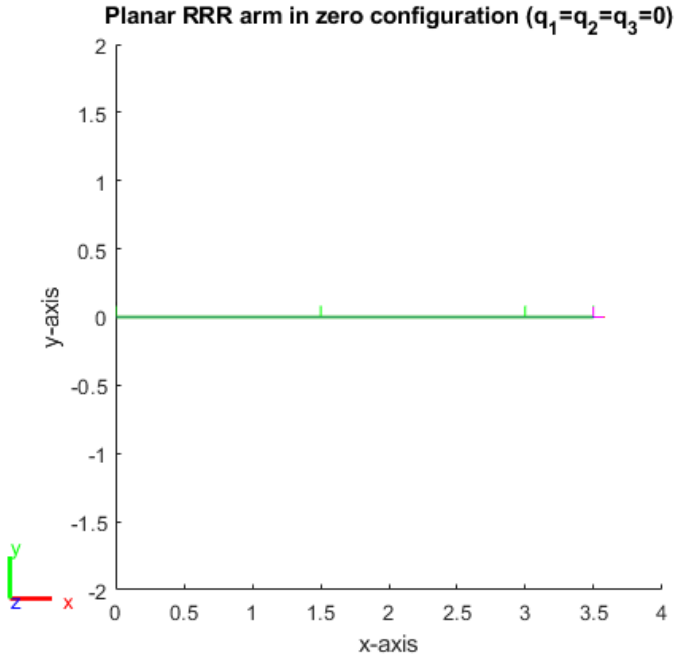

Figure 2: Zero Configuration of the Three-link planar robot arm.

2.1 PATH REPRESENTATION

In order for the end effector to trace the path, I first discretize the path and then find the normal component of each segment. Figure 2.11 is a plot of the path S and normal vectors along the path.

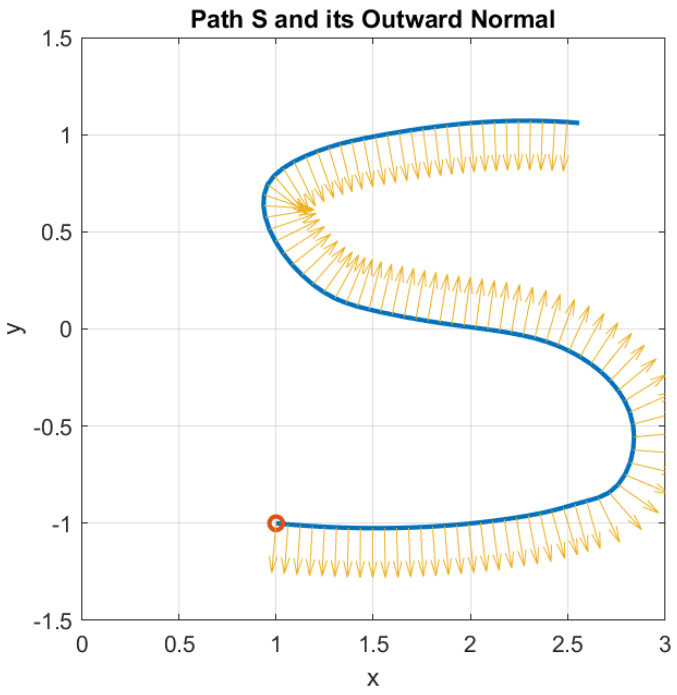

Figure 2.11: Path and its Normal Vectors

Next, I generated the path length array λ, and plotted it against the end effector position xT and yT in Figure 2.12.

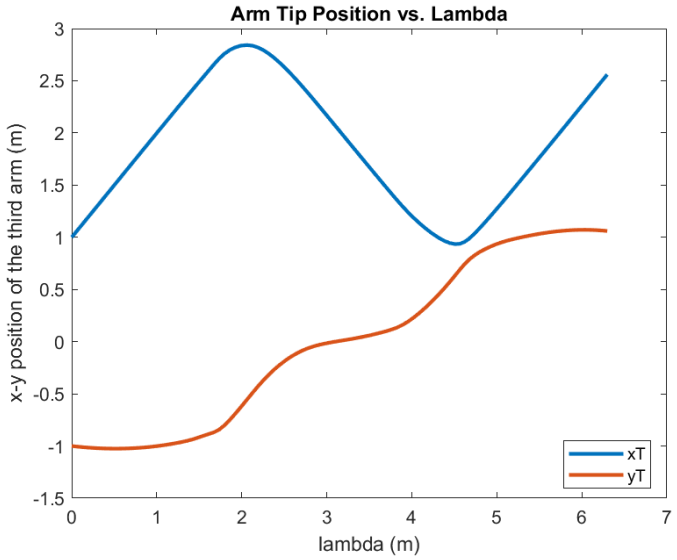

Figure 2.12: Tip Position and Lambda

2.2 Forward / Inverse Kinematics Theory

Forward Kinematics
In the three-lined robot arms, if we know the length of links (l_1, l_2, l_3) and the angle between each link (q_1, q_2, q_3), forward kinematics helps us to obtain the position p_{0T} and orientation q_{0T} of the end effector as following:

$$p_{0T} = \begin{bmatrix} l_2 \cos(q_1 + q_2) + l_1 \cos(q_1) + l_3 \cos(q_1 + q_2 + q_3) \\ l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) + l_3 \sin(q_1 + q_2 + q_3) \end{bmatrix}$$
$$q_{0T} = q_1 + q_2 + q_3$$

Inverse Kinematics
In this project, the results of the forward kinematics are the given, and the goal is to calculate the joint angles (q_1, q_2, q_3). In this case, inverse kinematics is used.

Algebraic Solution
With the following known:

$$p_{0T} = \begin{bmatrix} xT \\ yT \end{bmatrix}$$
$$q_{0T} = q_1 + q_2 + q_3$$

one can obtain:

$$xT - l_3 \cos(q_{0T}) = l_2 \cos(q_1 + q_2) + l_1 \cos(q_1)$$
$$yT - l_3 \sin(q_{0T}) = l_2 \sin(q_1 + q_2) + l_1 \sin(q_1)$$

squaring both side one can get:

$$l_1{}^2 + 2\cos(q_2)\, l_1\, l_2 + l_2{}^2 = (xT - l_3 \cos(q_{0T}))^2 + (yT - l_3 \sin(q_{0T}))^2$$
$$\cos(q_2) = -\frac{l_1{}^2 + l_2{}^2 - (xT - l_3 \cos(q_{0T}))^2 - (yT - l_3 \sin(q_{0T}))^2}{2\, l_1\, l_2}$$

As $\cos(q_2)$ can be obtained with two angles at max, one can then solve for the elbow up and elbow down solution.

Geometric Solution
Looking at this problem geometrically, the position and orientation of the end-effector is fixed at each segment points on the curve. Therefore we can rearrange the position kinematics equation to the following:

$$R_{10}(p_{0T} - p_{01} - R(qT)p_{3T}) = p_{12} + R_{12}p_{23} \tag{2.1}$$

where $p_{0T} - p_{01} - R(qT)p_{3T} = p_{13}$.
Now two unknowns ($q1$ in R_{10} and $q2$ in R_{12}) remain in the equation. We can visualize the geometry in Figure 2.21:

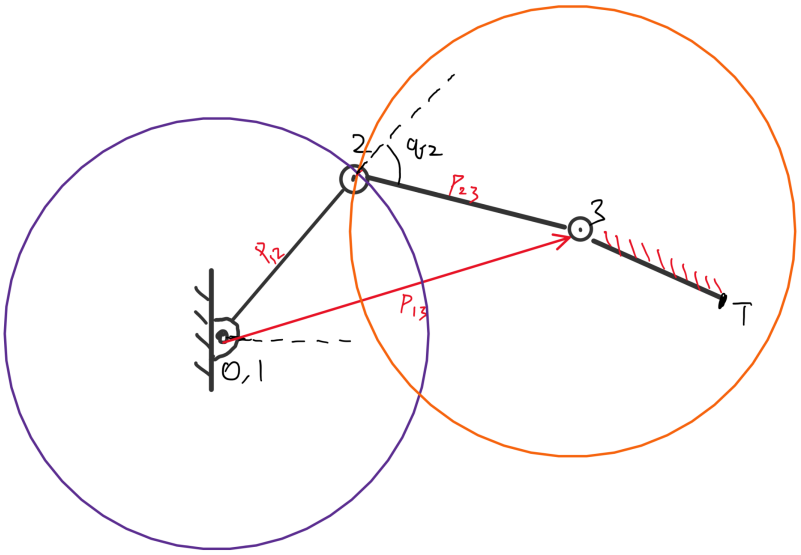

Figure 2.21: Geometric Representation

With p_{13}, p_{12}, p_{23} known, we can obtain angle $q2$ which gives us R_{12} using the Law of Cosine. Next, by plugging R_{12} back in to Equation (2.1), we can obtain R_{10} or $q1$. Lastly,

$$q3 = qT - q1 - q2$$

Note that sometimes, there will be two solutions as indicated in Figure 2.21.

Iterative Solution
Another approach for this problem is to look at the jacobian matrix that map joints angular velocity to the end-effector rate of change in the global frame. The iterative approach looks at inverse kinematics as a nonlinear least square problem as following:

$$\min_q \frac{1}{2} \left\| f(q) - \mathcal{X}_d \right\|^2, \quad \mathcal{X} = \begin{bmatrix} q_T \\ p_{0T} \end{bmatrix}$$

with derivative of cost function:

$$(f(q) - \mathcal{X}_d)^T J(q)$$

we can use gradient descent with the step size:

$$q_{k+1} - q_k = \alpha_k J^T(q_k)(f(q) - \mathcal{X}_d)$$

to iterate through the feasible domain.

2.3 Validation of Forward / Inverse Kinematics

Before applying the kinematics directly onto the path, we first validated the accuracy and compatibility's of the method. We start with a set of randomly generated joint angles

$$q_{rand} = [q1, q2, q3],\ qi \in [-\pi, \pi]$$

The robot was first generated with respect to this configuration. We then apply forward, inverse kinematics respectively using q_{rand}. As forward and inverse kinematics are inverse operation of each other, we expect the robot to have the exact same posture as the first configuration. What's more, we will have another solution with the same end-effector position.
The result was as expected:

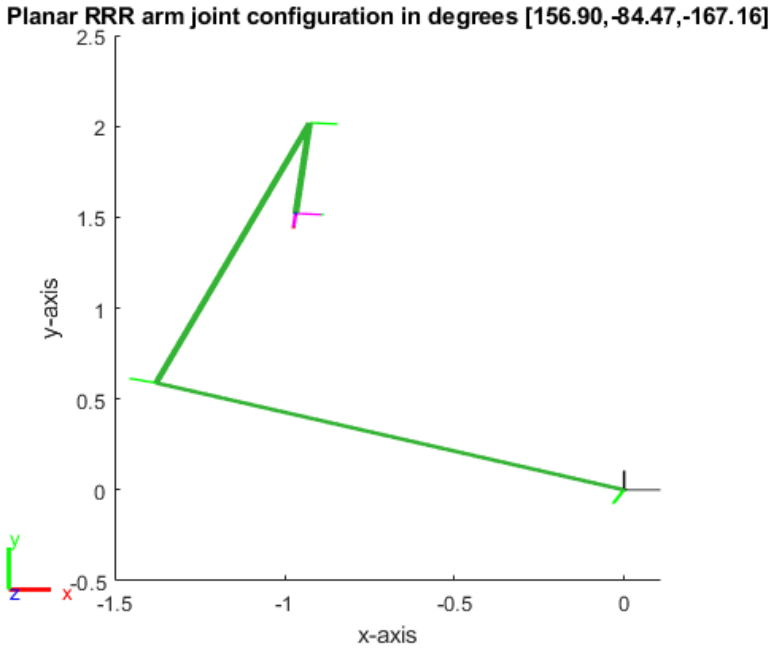

Figure 2.31: Random Robot Configuration

we used the geometric solution for the inverse kinematics:

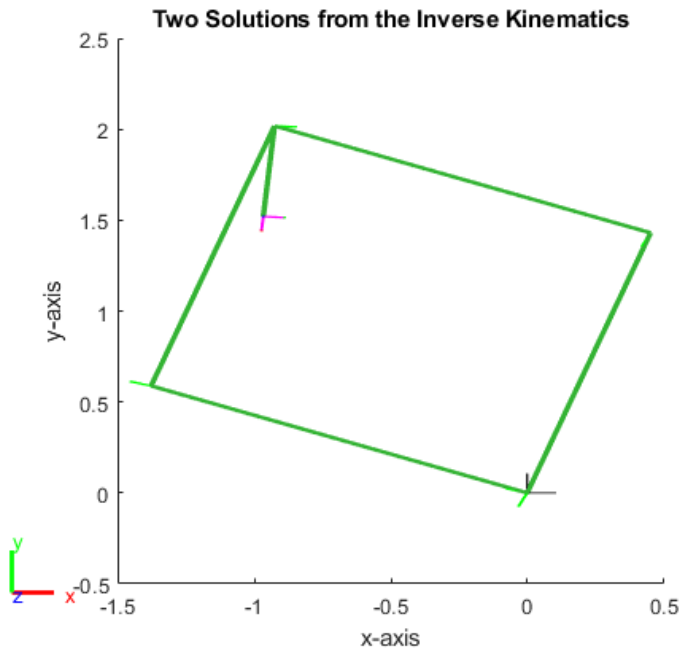

Figure 2.31: Random Robot Configuration

2.4 Tracing the Path

After confirming the code, the task is to generate the pose configurations of the arms along the entire path. We do so by changing the homogeneous transform function T of the robot based on the path information. The inverse kinematic function the convert the T to joint configurations q. The robot can then place its end-effector at the corresponding location. The solutions of the path is the following:

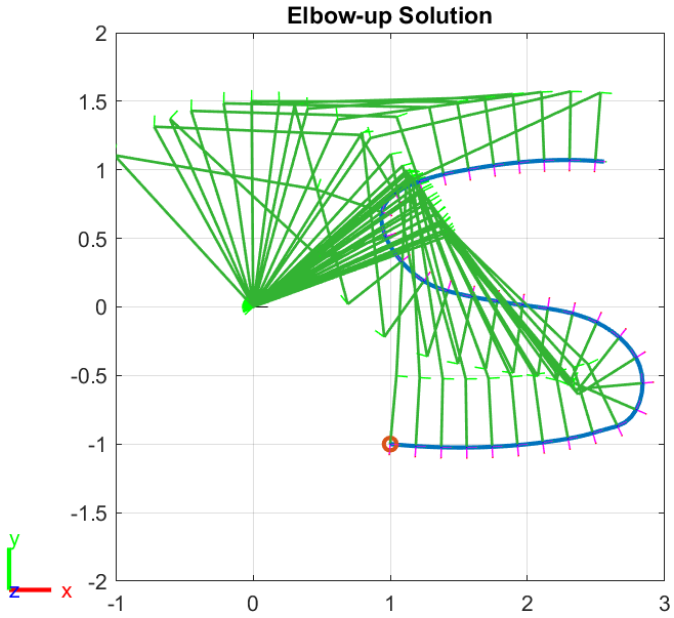

Figure 2.41: Elbow-up Pose Along the Path

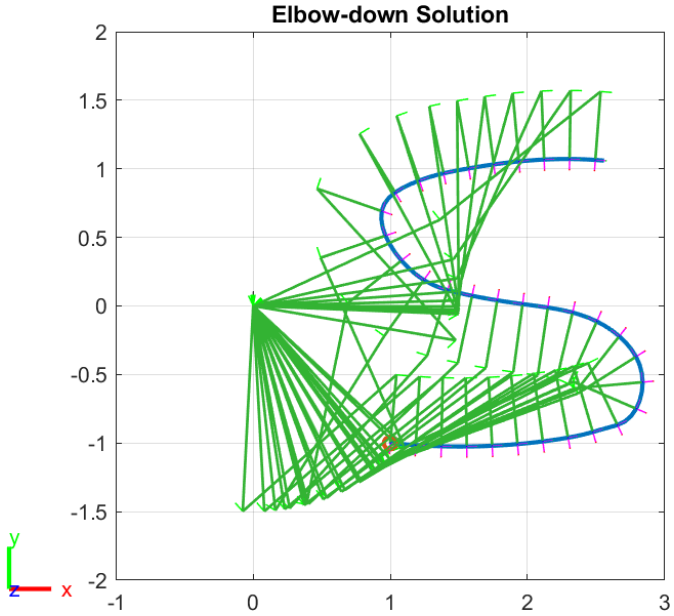

Figure 2.42: Elbow-down Pose Along the Path

2.5 Tracing the Path with Joint Speed Limit

Previously we have explored methods that maps joint angles to end-effector positions (forward kinematics) and maps end-effector back to joint angles (inverse kinematics). In reality,

there is always a speed limit on each joint that constraints the speed from one position to another. Specifically, when certain joint has higher speed than the other at that instance. With maximum robot joint velocities all 1 rad/sec, we can graph the maximum allowable speed for following:

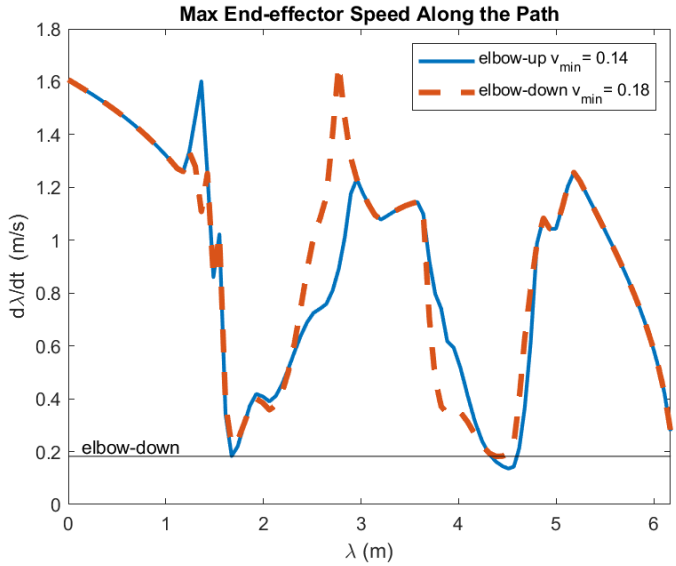

Figure 2.5: Path Velocity Determination

In Figure 2.5 $d\lambda/dt$ was calculated with:

$$\frac{d\lambda}{dt} = \frac{dq/dt}{dq/d\lambda}$$

When we increase the length of the arm however, the maximum possible end-effector speed decreased accordingly:

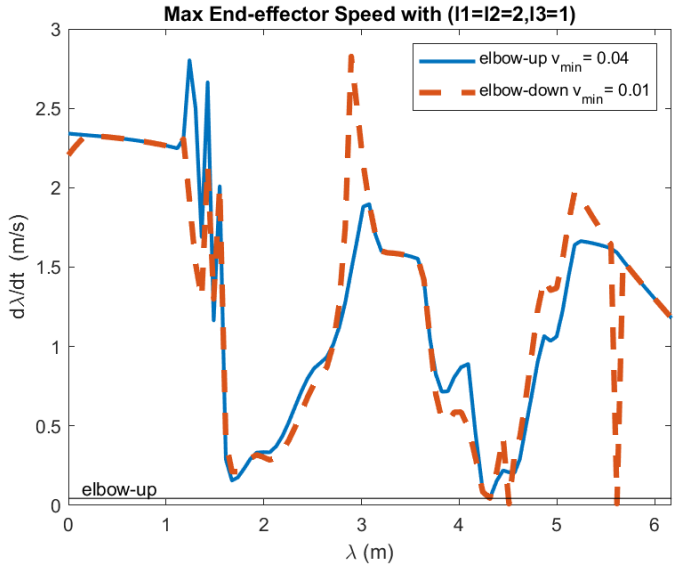

Figure 2.51: Path Velocity with Longer Arm Length

When we move the path S away from the root of the robot arm, although the upper bound increases, the smallest value also decreases as it get closer to singularity towards the end of the path.

2.6 Obtaining a Path Velocity Profile with Jacobian based Method

Jacobian method allow us to create a mapping between joint speed and the end-effector rate of change. Such method enable us to create a varying $d\lambda/dt$ profile which in theory would increase the overall speed of end-effector swiping across λ.
Suppose

$$A = \begin{bmatrix} \frac{dq}{d\lambda} \\ \frac{dx}{d\lambda} \\ \frac{dy}{d\lambda} \end{bmatrix}, \frac{dq}{dt} = \begin{bmatrix} \frac{dq_1}{dt} \\ \frac{dq_2}{dt} \\ \frac{dq_3}{dt} \end{bmatrix}$$

$$A * \frac{d\lambda}{dt} = J * \frac{dq}{dt}$$

$$\frac{d\lambda}{dt} = A^{-1} J \frac{dq}{dt}$$

We set this up as a linear programming problem to:

$$\min\ A^{-1} J \frac{dq}{dt},\ -1 \leq \frac{dq}{dt} \leq 1$$

where the optimization variable is $\frac{dq}{dt}$.
The result is the following:

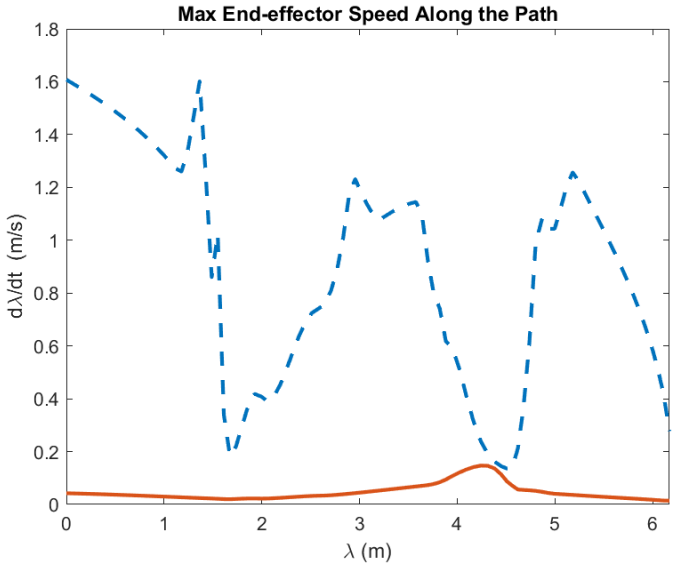

Figure 2.61: Elbow-up Jacobian Optimized Solution

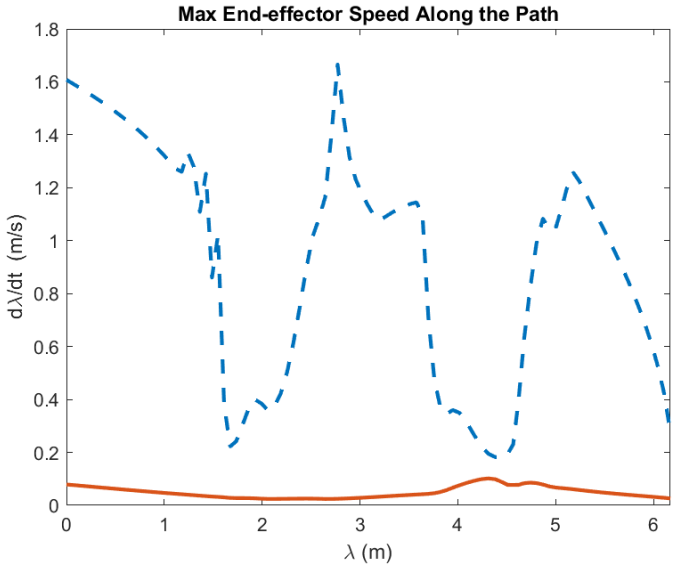

Figure 2.61: Elbow-down Jacobian Optimized Solution

The shown result was acceptable but not precise. The expected result was for the velocity profile to be closely tracking the maximum possible boundary. The result however seems to be over constrained. After several rounds of sanity check, an alternative optimization strategy will be tested in the future to further improve the result.

3 Conclusion

In this mini-project, we studied the kinematics theory of the 3-link planner robotics arm. Key takes away is how the numbers of links relates to the capability of the robot.
A remark was mention in class that industrial robot arms typically consists of 6 degree of freedom. With the knowledge of jacobian, I understand that provided with X, Y, Z, ω_x, ω_y, ω_z, six degree of freedom allow the end-effector to achieve most positions without redundancy.
Another take away is that higher DOF robot usually uses optimization algorithms to perform inverse kinematics.