MEAM 520 Lab 3

Raima Sen, Renu Reddy Kasala

November 5, 2022

1 Introduction

This report presents the complete derivation of the Jacobian matrix for the FRANKA EMIKA PANDA arm. This Jacobian matrix is further used to derive the forward kinematic velocity and inverse kinematic velocity equations to calculate the end effector velocity and the joint velocities of the manipulator arm, respectively.

2 Methodology

2.1 Calculating the Jacobian Matrix (J)

The Jacobian matrix is split into the linear velocity Jacobian and the angular velocity Jacobian. The following methodology was adopted to calculate the Jacobian matrix of the FRANKA EMIKA PANDA arm:

- From Lab 1 we had obtained the intermediate joint positions and transformation matrix of the end effector w.r.t frame 0. For this lab, we required both the position and the orientation (essentially the pose) of each joint w.r.t frame 0.
  The homogeneous transformation matrices are as follows :

$$T_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0.141 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_2 & -c\theta_2 & 0 & 0.192 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ 0 & 0 & 1 & -0.195 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -c\theta_4 & s\theta_4 & 0 & 0.121 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_5^4 = \begin{bmatrix} 0 & 0 & -1 & -0.125 \\ c\theta_5 & -s\theta_5 & 0 & -0.0825 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_6^5 = \begin{bmatrix} -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & -1 & 0.015 \\ c\theta_6 & -s\theta_6 & 0 & 0.259 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_7^6 = \begin{bmatrix} 0 & 0 & -1 & -0.051 \\ -c\theta_7 & s\theta_7 & 0 & -0.088 \\ s\theta_7 & c\theta_7 & 0 & 0.015 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_e^7 = \begin{bmatrix} 0.707 & 0.707 & 0 & 0 \\ -0.707 & 0.707 & 0 & 0 \\ 0 & 0 & 1 & 0.159 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  Using **Brute Force Method**, in Lab 1, we had computed the intermediate joint positions (7 3x1 vectors containing the x, y and z positions of each joint) along with the E.E. homogenous transformation matrix.
  For this Lab, we shall be required to calculate the complete transformation matrix of each joint w.r.t frame 0. i.e.

$$T_1^0, T_2^0, T_3^0, T_4^0, T_5^0, T_6^0, T_7^0$$

  We shall also require the end effector position w.r.t the base frame T_e^0 for particularly calculating the linear velocity Jacobian.

- Calculating Linear Velocity Jacobian (J_v)
  Among the 2 methods - Analytical and geometric, we adopted the geometric approach for finding the linear velocity Jacobian. Since all joints of the FRANKA arm are revolute, the following equation is used to calculate J_v :

  $$J_{v_i} = \hat{z}_{i-1} \times (\vec{o_n} - \vec{o_{i-1}})$$

  Here all vectors are expressed in frame 0. For each joint 'i', the third column of each joint's transformation matrix provides $\hat{z}_{i-1}$. $\vec{o_n}$ is uniform and is the final position of the end effector expressed in the base frame.$\vec{o_{i-1}}$ for each joint is the position of each joint expressed in the base frame. It is obtained from the fourth column of each intermediate joint's transformation matrix that we obtained in the previous step. The J_v matrix is a (3xn) matrix, where n is the number of joints. For the FRANKA arm, we have 7 joints so J_v is a (3x7) matrix.

- Calculating Angular Velocity Jacobian (J_w)
  For the angular velocity jacobian, we used the following equation :

  $$J_{\omega_i} = \hat{z}_{i-1}$$

  Each joint's contribution the the end effector's rotational velocity would be the unit vector of the rotational matrix in the z direction for the ith joint expressed in the base coordinate frame. Since the panda arm has all revolute joints, $\rho = 1$ in $J_\omega = [\rho_i \hat{z}_{i-1}]$. The franka arm has 7 joints so the final J_ω matrix will be a (3x7) matrix. We acquire z_{i-1} from our implementation for the forward kinematics in Lab 1.

- Stacking the J_v and J_ω we get a (6x7) matrix constituting the complete Jacobian matrix for the FRANKA arm.

2.2 Calculating End Effector Velocity (v) from Forward Kinematic Velocity

For finding the end effector velocity, we use the following equation :

$$\vec{v} = J(\vec{q})\dot{q}$$

$$\text{Where, } \vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \quad \dot{q} = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{q_5} \\ \dot{q_6} \\ \dot{q_7} \end{bmatrix}$$

The Jacobian matrix was used from the previous section at a given configuration q. The dimensions of J is (6x7), the dimensions of $\dot{q}$ is (7x1) as there are 7 joints in the arm. This provides the full velocity vector (6x1) vector representing the linear and angular velocities in the x,y, and z directions.

2.3 Calculating Joint Velocities ($\dot{q}$) from Inverse Kinematic Velocity

The Jacobian of the FRANKA panda arm is non square, as is evident in section 2.1. It is a (6x7) matrix as the arm has 7 joints. Therefore for a redundant system like the panda arm,

$$rank(J) <= min(N, 6)$$

To calculate the joint velocities, we shall require to take the inverse of J, which for this case translates to a pseudoinverse. The input for this would be the end effector linear and angular velocities at a given configuration of the robot. The output would be all the possible joint velocities to achieve the end effector velocity.
If a particular velocity component of the end effector is unconstrained (for example, if the $\vec{v_x}$ is not

constrained, then the first row of the J matrix is eliminated, which results in a (5x7)matrix and so on). For the underdetermined FRANKA arm,there can either be a condition where :
a) There are no solutions under singular configuration
b) There are infinitely many solutions
For case a) since there are no exact solution, the closest solution which obtains the end effector velocity is chosen as the solution. For case b) since there are multiple solutions, the solution that gives the minimum L2 norm between the target and input end effector velocity is the optimal solution. Mathematically,

$$\dot{q} = J^{+}\xi$$

Where ξ is the desired velocity.
For case a) $\dot{q}$ gives the least square error. Mathematically,

$$\dot{q} = argmin||\xi - J\dot{q}||^2$$

For case b) when there are infinitely many solitions the solution space looks like,

$$\dot{q} = J^{+}\xi + (I - JJ^{+})b$$

2.4 Code Structure

The code written by Raima Sen was used for experiments. The transformation matrices are defined in the lab3_data function of FK class from Lab 1. Using post multiplication the final transformation matrix of the end effector in the base frame is obtained. The intermediate joint poses are obtained by doing post multiplication in a step by step manner so that each joint's transformation matrix can be obtained with respect to the base frame.
the lab3_data function is called to calcJacobian.py and the corresponding J_v and J_ω are iteratively calculated for each joint. This Jacobian matrix is further used for finding the forward and inverse kinematic velocities. For the forward kinematic velocity a dot product is taken between the Jacobian and the joint velocities vector to get the end effector velocity. Using the least squares approach available in np.linalg.lstsq, all possible solutions of the joint velocities was obtained.

3 Evaluation

We tested our code on simulation and on the hardware and found that the arm was behaving exactly as predicted. Following are the images of Simulation vs Hardware:

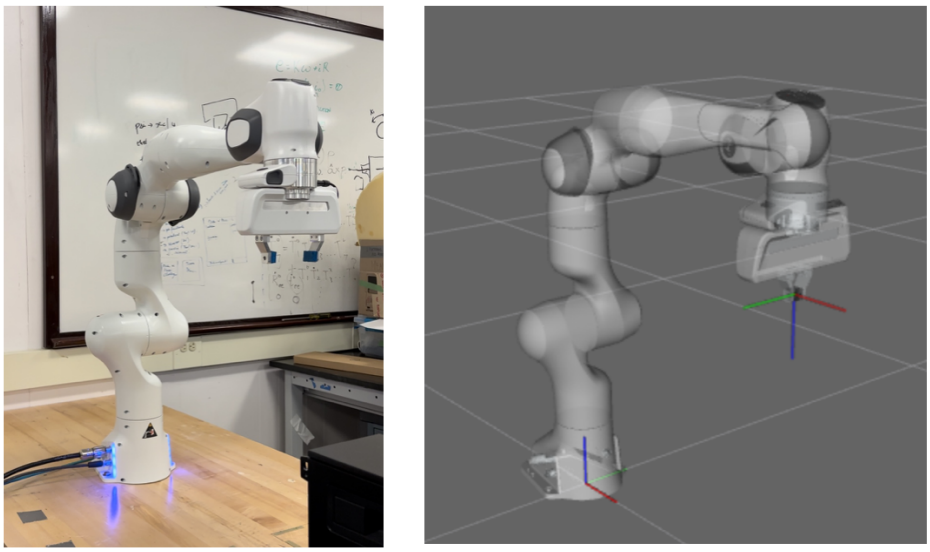

Figure 1: Neutral position $[0, 0, 0, -\pi/2, 0, \pi/2, \pi/4]$ in hardware and simulation

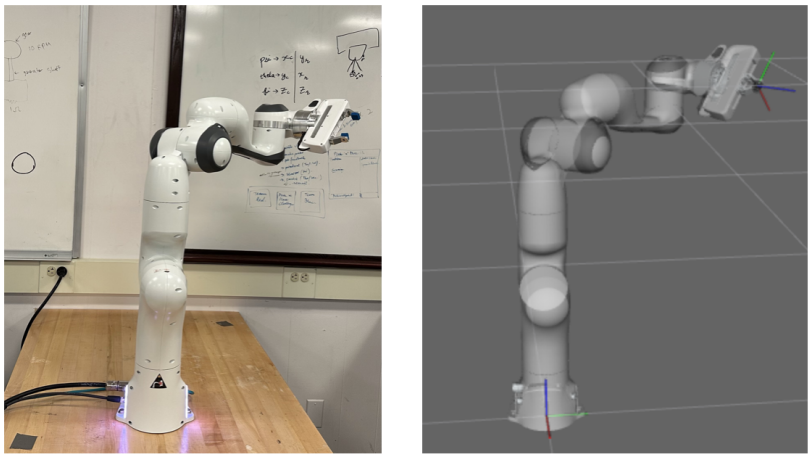

Figure 2: Configuration 1: $[\pi/2, 0, \pi/4, -\pi/2, -\pi/2, \pi/2, 0]$ in hardware and simulation

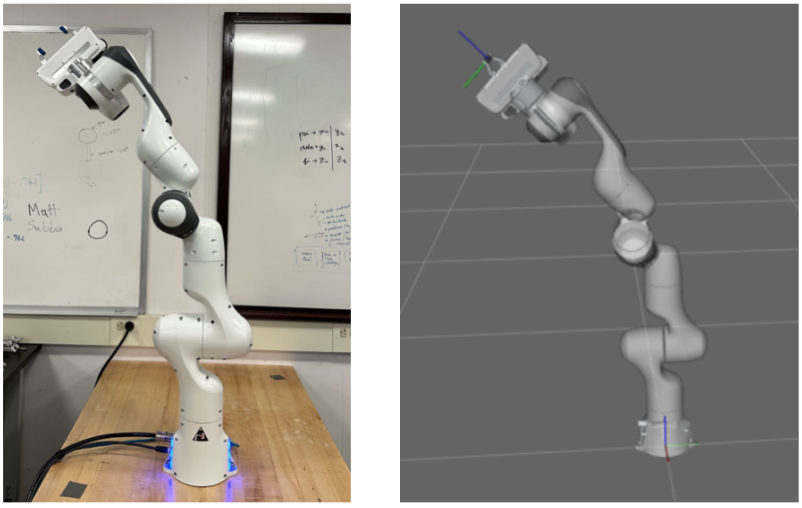

Figure 3: Configuration 2: $[0, 0, -\pi/2, -\pi/4, \pi/2, \pi, \pi/4]$ in hardware and simulation

To evaluate the correctness of our code we started in the zero configuration, and assumed that only one joint θ_6 moves by 90^0. The corresponding linear velocity of the end-effector is $\vec{v} = J(\vec{q})\dot{q}$

Linear Velocity $\vec{v} = \begin{bmatrix} 0.788 * \dot{\theta}_2 - 0.472 * \dot{\theta}_4 - 0.088 * \dot{\theta}_6 \\ 0.21 * \dot{\theta}_1 + 0.21 * \dot{\theta}_3 + 0.21 * \dot{\theta}_5 \\ -0.21 * \dot{\theta}_2 + 0.1275 * \dot{\theta}_4 + 0.21 * \dot{\theta}_6 \end{bmatrix}$

Since there is no motion in all angles except for θ_6.We got:

$\vec{v} = \begin{bmatrix} -0.088\dot{\theta}_6 \\ 0 \\ -0.21\dot{\theta}_6 \end{bmatrix}$

From geometry, we can see that the velocity vector of the end-effector should point in negative x and positive z direction, which is the same as that of the one obtained through code. Velocity direction can be obtained by drawing tangent at that point. To verify the magnitude of linear velocity:

From the Figure 4(b) we get:

$\alpha = \theta_6$ - $tan^{-1}\frac{d_7}{a_6}$

$V_x = \text{rcos}(\alpha)$; $V_y = \text{rsin}(\alpha)$

$\vec{v} = \vec{w}$ x r

$\vec{v} = \begin{bmatrix} 0 \\ 0 \\ w \end{bmatrix} * \begin{bmatrix} rcos(\alpha) \\ rsin(\alpha) \\ 0 \end{bmatrix}$

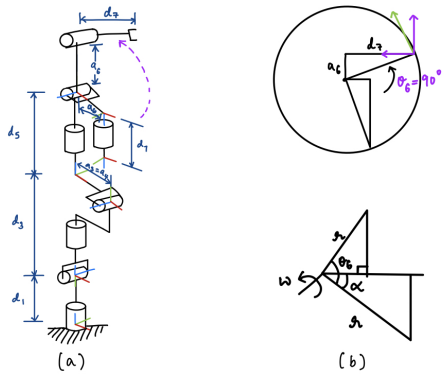

Figure 4: (a):schematic diagram depicting movement of joint 6. (b)Geometric visualisation of θ_6

Then the linear velocity of the end effector is given by $V_e^0 = R_6^0\ V_e^6$, where e denotes end effector.

$$V_e^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \text{-wrsin}(\alpha) \\ 0 \\ \text{wrcos}(\alpha) \end{bmatrix} = \begin{bmatrix} -\dot{\theta_6}(0.088) \\ 0 \\ -\dot{\theta_6}(0.21) \end{bmatrix}$$

This is the same as the one obtained from $\vec{v} = J(\vec{q})\dot{q}$. FK reflects this velocity. The rate of change of position, which is the linear velocity has the same signs, which confirms the correctness of the linear velocity.

Using geometric intuition, we can figure out where there are singularities. We can see that these configurations reflected in our mathematical expressions at conditions where the determinant of the Jacobian is zero. For singularity configuration: There are two types. One is parametric singularity. It is when any of the dh paramaters become zero. The other is configuration singularity. It is when $\det(J_v) = 0$, i.e when jacobian loses rank or jacobian columns/rows become linearly dependent. In other words, the robotic arm loses one or more degrees of freedom at certain points in the configuration space where infinitesimal motion in a particular direction is not possible.

Also since is mathematically challenging to compute determinant of Jacobian for Franka panda arm, we can decouple the determination of singular configurations into wrist and arm configurations.

For the wrist configuration (refer Figure 5), the matrix will be singular when the two wrist axes align(Like discussed in Lecture slides 11VelocityIK). That is when joints 1 and 3 become coincident, which corresponds to $\theta_2 = 0$ or π.

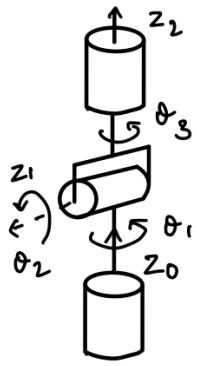

Figure 5: Spherical wrist

One of the singular configuration zero configuration q =[0,0,0,0,0,0,0]. Here, the velocity jacobian is:

$$J_v = \begin{bmatrix} 0. & 0.49 & 0. & -0.174 & -0. & 0.21 & 0. \\ 0.088 & 0. & 0.088 & 0. & 0.088 & 0. & 0. \\ 0. & -0.088 & 0. & 0.005 & 0. & 0.088 & 0. \\ 0. & -0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & -1. & 0. & -1. & -0. \\ 1. & 0. & 1. & 0. & 1. & 0. & -1. \end{bmatrix}$$

The columns 1, 3 and 5 are the same. That is, they are linearly dependent. Hence, this is one of the singularity configurations.

To check the correctness of the Forward Velocity code, we used Visualize.py to visually check the linear and angular velocities of the endeffector as each joint was rotated. Using the right hand thumb rule we could verify the directions of angular and linear velocities.

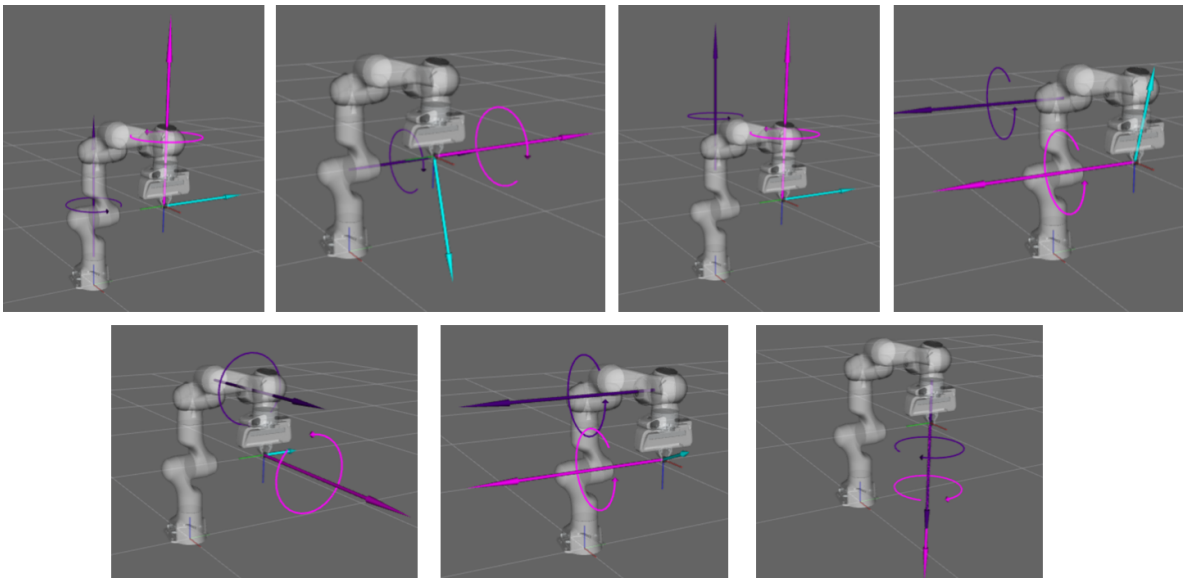

Figure 6: Neutral Configuration:$[0, 0, 0, -\pi/2, 0, \pi/2, \pi/4]$ with all joint from 1-7 in consideration from left to right

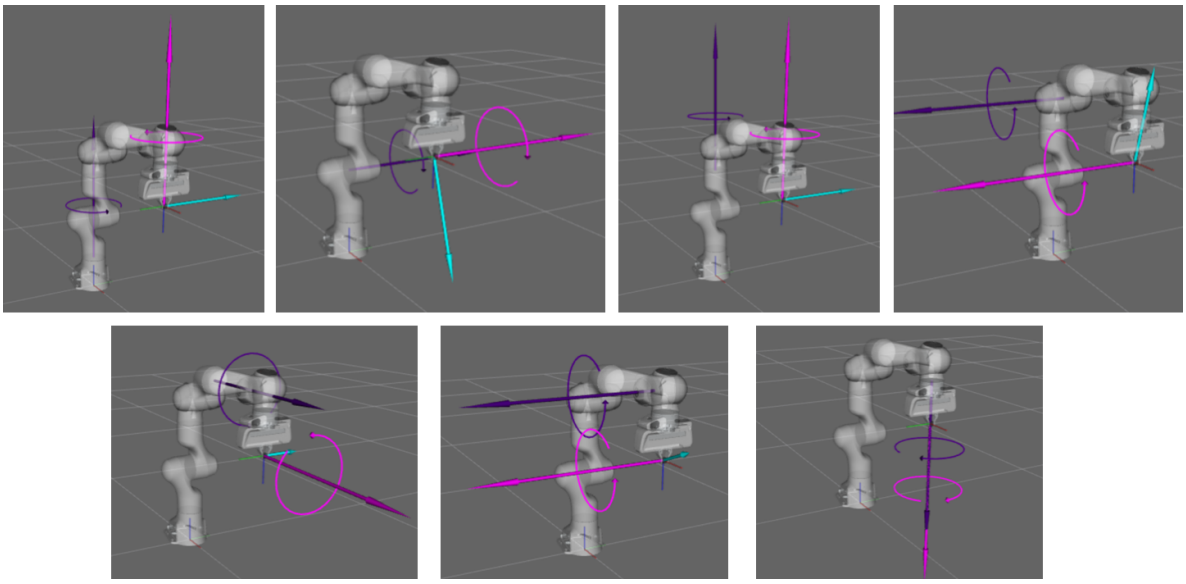

Figure 7: Configuration 2:$[0, 0, -\pi/2, -\pi/4, \pi/2, \pi, \pi/4]$ with all joint from 1-7 in consideration from left to right

To check the correctness of the Inverse Velocity code we did the following:
If the end effector solution exists for the joint velocities it would have infinitely many solutions. So, doing manually verifying of our inverse velocity code is not possible. If we use certain joint velocity and a Jacobian to calculate the end effector velocity that might not be same as the joint velocity calculated from the inverse kinematics. This is because there exists infinitely many solutions of joint velocities. Hence we did a visual interpretation of our solution using follow.py. So we monitored the ellipse, eight and line trajectories and checked if the robot performs expected behaviour. Our end effector tracked trajectories as expected.
For the ellipse trajectory: $x_{des} = x_0 + [(r_y * sin(f * t), r_z * cos(f * t), 0)]$
where t = time in sec since start
f = frequecny in rad/s of the trajectory
ry, rz = radius in m of the x and y portion
x_0 = initial position of the end effector
For the line trajectory: $x_{des} = x_0 + [f * L * t, f * L * t, 0]$
where l = length of the line in meters
And x_{des} =target end effector position in the world frame
v_{des} = target end effector linear velocity in the world frame. This is the derivative of the position. Further we tested the eight trajectory on the simulator as well as hardware.

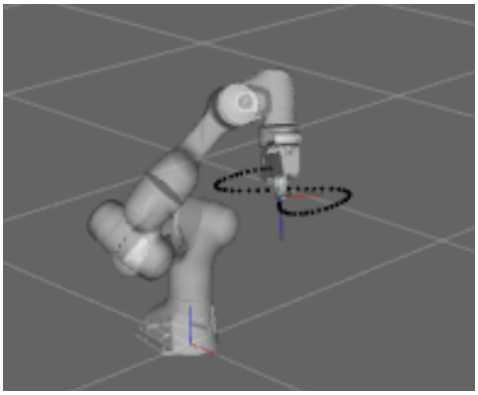

Figure 8: End effector tracking eight shape trajectory

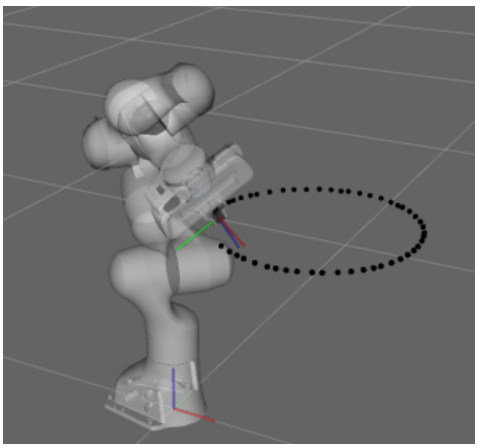

Figure 9: End effector tracking ellipse shape trajectory

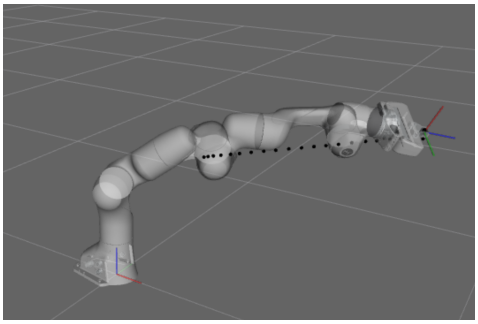

Figure 10: End effector tracking line shape trajectory

4 Analysis

- Using the forward velocity kinematics we get the velocity of the end effector. And by using this velocity in inverse velocity kinematics we'll get the joint velocities $\dot{q}$. And this is the same as the one we started with in Forward velocity kinematics. Doing this we can confirm the correctness of our methodology. Also, we should make sure we are avoiding singular configurations.
- Proportional control: When Kp =20, the arm in majority of the cases is unable to follow the given trajectory since the robot joint limits are exceeded, the velocity is too fast. For a smaller value of Kp, the output signal will become smaller. So the system would be slower to respond. And if the value is too less then the output signal is so small that it stalls. If the Kp value is large, then the output signal becomes larger so reaches the goal value faster. However, too huge Kp can lead to velocity being too fast, that it exceeds the limits. So it is essential to find the optimal value of Kp to reach the goal at an optimal pace. We used Kp = 2.
- The Franka panda arm is good at movements like circle, ellipse, eight trajectories. It is also good at line motion within the joint limits. But it is bad when there is a sharp change in velocity like in the case of polygon trajectories. Further, discontinuous trajectories will also be poorly performed by the arm.
- The Franka panda arm would have issues traversing with given velocities when close to singularities. Like mentioned in [1], at singularities, bounded end-effector velocities may correspond to unbounded joint velocities.
- Position control is required for tasks where the manipulator is not interacting significantly with objects in the workplace. For example in pick and place tasks, material transfer, spot welding etc. And we would want to use velocity control over position control in applications like spray painting where we can change the speed as per the desired application requirement. It is also useful in cases where the endeffector need to maintain constant velocity.

Here, is the link to the hardware testing we did- Franka Panda arm following an eight trajectory or please go to the url `https://youtube.com/shorts/951r88WANVg?feature=share`

[1] Hutchinson, Seth A. and Mathukumalli Vidyasagar. "Robot modeling and control / Mark W. Spong, Seth Hutchinson, M. Vidyasagar." (2006).