ME 639
Introduction to Robotics

Mid Semester Project

OSAKE I

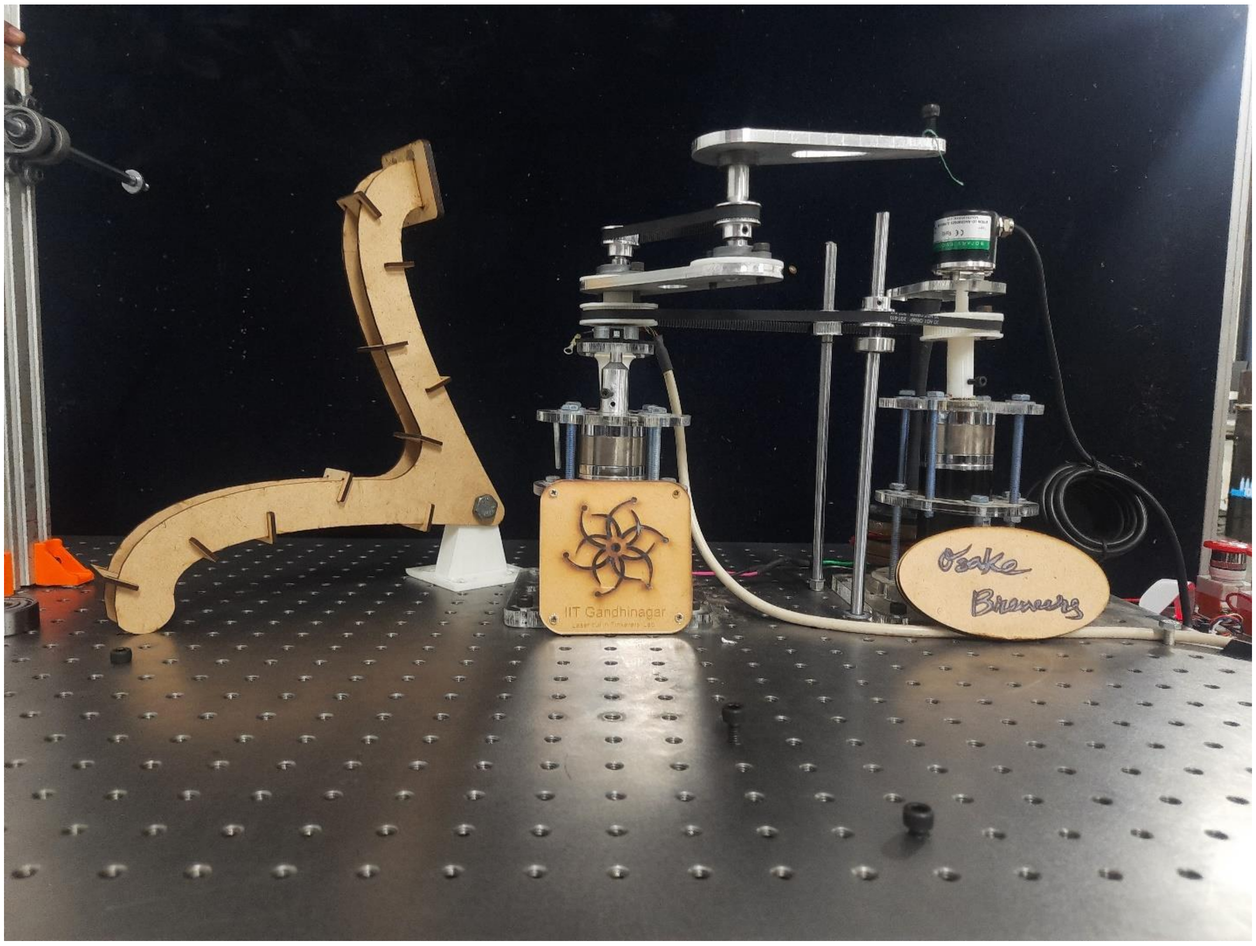
Submitted By – OSAKE Brewers
Souritra Garai (18110166)
Krunalkumar Patel (22210022)
Sunil Kumar (20250042)
Rishab Kumar (22250029)
Avnish Chokshi (22210009)
Bandaru Goutham (22210016)
Ishrath (22270002)

Indian Institute of Technology

Gandhinagar

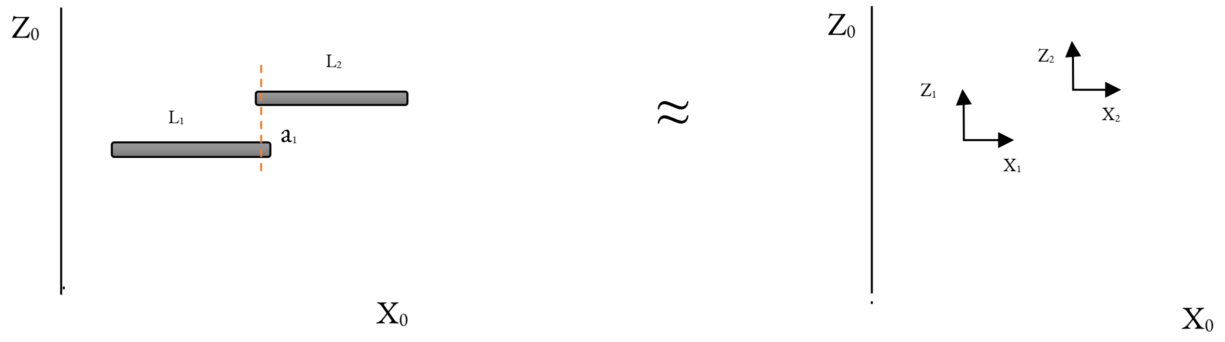
Semester I 2022-23

1.1 Robot Image



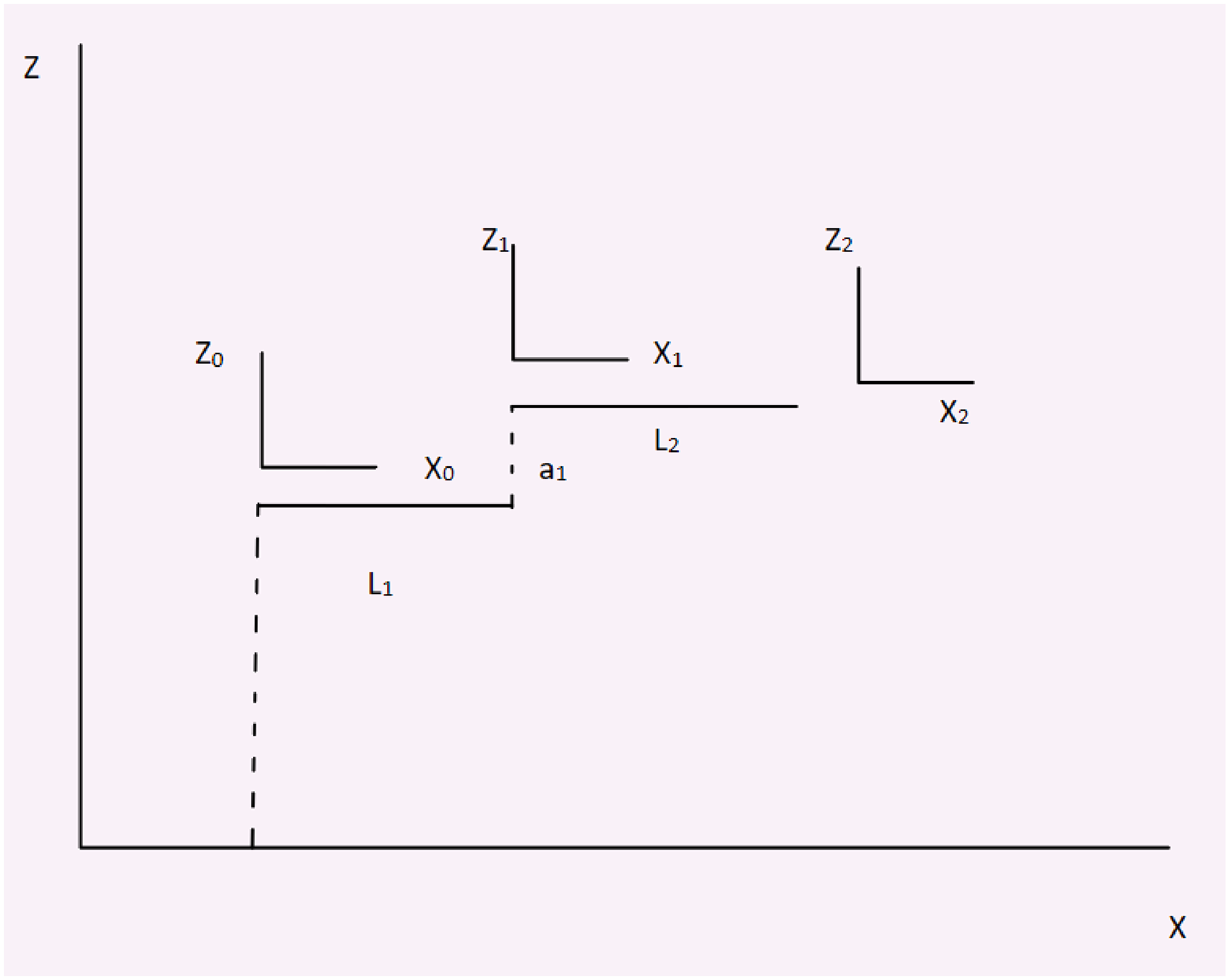
1.2 Task 0 - Forward Kinematics

The D-H Parameters of the manipulator can be computed by assuming the co-ordinate frames as shown below –



Links	**α**	**ϴ**	**a**	**d**
Link 1	0	θ_1	L_1	a_1
Link 2	0	θ_2	L_2	0

DH Parameters, Jacobian, end effector velocity and position of OSAKE-1



Link	α	a	θ	d
1	0	L_1	θ_1	a_1
2	0	L_2	θ_2	0

$$T_0{}^2 = T_0{}^1 T_1{}^2$$

$$T=\begin{bmatrix} C_1 & -S_1C_\alpha & S_1S_\alpha & aC_1 \\ S_1 & C_1C_\alpha & -C_1S_\alpha & aS_1 \\ 0 & S_\alpha & C_\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0{}^2=\begin{bmatrix} C_{12} & -S_{12} & 0 & L_2C_{12}+L_1C_1 \\ S_{12} & C_{12} & 0 & L_2S_{12}+L_1S_1 \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

End effector Position can be computed as -

$$\begin{bmatrix} P_0 \\ 1 \end{bmatrix}=T_0{}^2\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}=\begin{bmatrix} L_2C_{12}+L_1C_1 \\ L_2S_{12}+L_1S_1 \\ a_1 \\ 1 \end{bmatrix}$$

Jacobian Matrix is computed as -

$$J=\begin{bmatrix} Z_0\left(O_2-O_0\right) & Z_1\left(O_2-O_1\right) \\ Z_0 & Z_1 \end{bmatrix}$$

Where, $Z_0, Z_1=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ O_0=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},\ O_1=\begin{bmatrix} L_1C_1 \\ L_1S_1 \\ a_1 \end{bmatrix},\ O_2=\begin{bmatrix} L_2C_{12}+L_1C_1 \\ L_2S_{12}+L_1S_1 \\ a_1 \end{bmatrix}$

$$J_{11} = \begin{bmatrix} -\left(L_2 S_{12} + L_1 S_1\right) \\ L_2 C_{12} + L_1 C_1 \\ 0 \end{bmatrix}$$

$$J_{12} = \begin{bmatrix} -L_2 S_{12} \\ L_2 C_{12} \\ 0 \end{bmatrix}$$

$$J = \begin{bmatrix} -\left(L_2 S_{12} + L_1 S_1\right) & -L_2 S_{12} \\ L_2 C_{12} + L_1 C_1 & L_2 C_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

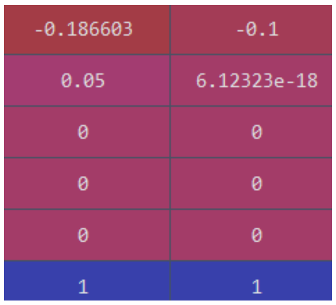
$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

The end effector velocity can be expressed as -

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\left(L_2 S_{12} + L_1 S_1\right) & -L_2 S_{12} \\ L_2 C_{12} + L_1 C_1 & L_2 C_{12} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

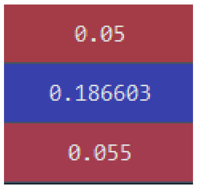
Sample code outputs -

For $q_1 = \pi/3$ and $q_2 = \pi/6$



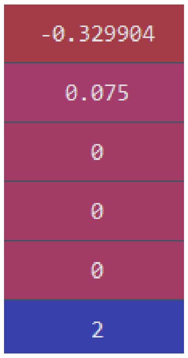
-0.186603	-0.1
0.05	6.12323e-18
0	0
0	0
0	0
1	1

Figure 3 Jacobian matrix



0.05
0.186603
0.055

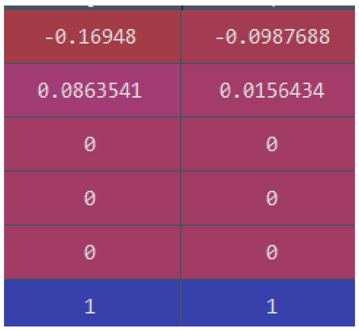
Figure 4 Position vector of end effector



-0.329904
0.075
0
0
0
2

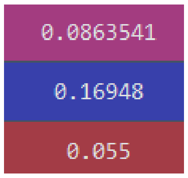
Figure 5 End effector velocity vector

For $q_1 = \pi/4$ and $q_2 = \pi/5$



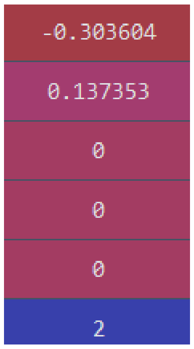
-0.16948	-0.0987688
0.0863541	0.0156434
0	0
0	0
0	0
1	1

Figure 6 Jacobian matrix



0.0863541
0.16948
0.055

Figure 7 End effector position vector



-0.303604
0.137353
0
0
0
2

Figure 8 End effector velocity vector

Task 1 - Trajectory

We received hardware with low level torque (motor current) controllers already implemented in place. To follow a trajectory, we need to divide the trajectory into discrete points in the state space and give the torques required to move from one point to the next. However, to calculate the torque we need to consider the dynamics of the manipulator. In the given system, there are a lot of unknowns that directly affect the manipulator dynamics, for example, moment of inertia of the gears in the motor's gearbox, the static friction (stiction) in the motor, etc. This makes modelling the dynamics rather cumbersome.

Hence, we implemented a closed loop PI (Proportional – Integral) controller to move from one point to the next. Initially we implemented the higher level trajectory controller at the same frequency as the torque controllers on the microcontroller, i.e., ≈ 2 kHz. But this led to a highly jittery motion of the manipulator. Next, we implemented a non-blocking 'if' structure in the code to slow down the trajectory controller to 100 Hz. This immediately smoothened out the motion of the robot.

We followed a circular and a linear trajectory with the end-effector of the manipulator. The trajectories were largely skewed due to static friction in the system with different values in different configurations of the robot.

Task 2 – Constant Force

The 2R manipulator setup is remotely driven using pulleys and belts. Thus, the two motors change the absolute angles with respect to the ground coordinate, at the joints. Hence, we used absolute angles and the corresponding Jacobian (not the one derived with D-H parameters) for our analysis.

The task was to bring the end-effector to a point on a wall and apply a constant force. We used a hinge mechanism to convert the horizontal force on the wall to a vertical force on a weighing scale. This helped us to easily measure the force by the end-effector.

When end-effector reached the end of the trajectory, the abrupt change in current (due to transition from trajectory controller to force controller) led to a disruptive response from the manipulator system. The smoothen out the change in current we used a simple low-pass digital filter upon Shail Jadav's recommendation. The abrupt motion of the manipulator decreased largely after using the digital filter.

In the force experiments, we hardcoded the microcontroller to apply a constant 2 N force after reaching the end of the trajectory. The code uses the current manipulator configuration to calculate the Jacobian and subsequently calculate the required joint torques. However, on the weighing scale, we were able to observe forces up to 130 g-wt. The errors may again be explained by the static friction in the system.

Task 3 – Spring

A force proportional to the deviation from virtual spring point was set as desired force at the end effector. The same methodology as Task 2 was followed – calculate the Jacobian from the current manipulator configuration and multiply its transpose with the desired force at the end point. Again, due to uneven static friction of the robot joints, the spring motion was largely damped. Further, the dampness in motion was different for different configurations and perturbations.