University Institute of Engineering

Department of Electronics & Communication Engineering

Experiment No. :- 8

Student Name: Priyanshu Mathur **UID:** 20BEC1073

Branch: Electronics and Communication **Section/Group:** A

Semester: 7th **Date of Performance:** 15/11/2023

Subject Name: Industrial Automation & Robotics **Subject Code:** 20ECA-446

1. **Aim of the practical:** Simulate two axis of robotics arm by LINK and Serial Link Object function of RTB 10.4 using MATLAB.

2. **Tool Used:** MATLAB

3. **Theory:-**

This MATLAB code utilizes Robotics Toolbox 10.4 to simulate a 6-DOF robotic arm. DenavitHartenberg parameters in Link objects define joint properties, creating a SerialLink object for kinematic representation. The plot function visualizes the arm's initial configuration. Using mdl_puma560 and teach enables an interactive interface for the Puma 560 arm, allowing users intuitive control in a virtual environment. This comprehensive approach ensures accurate simulation and analysis of the robotic arm's geometric and kinematic properties, facilitating effective research and development.

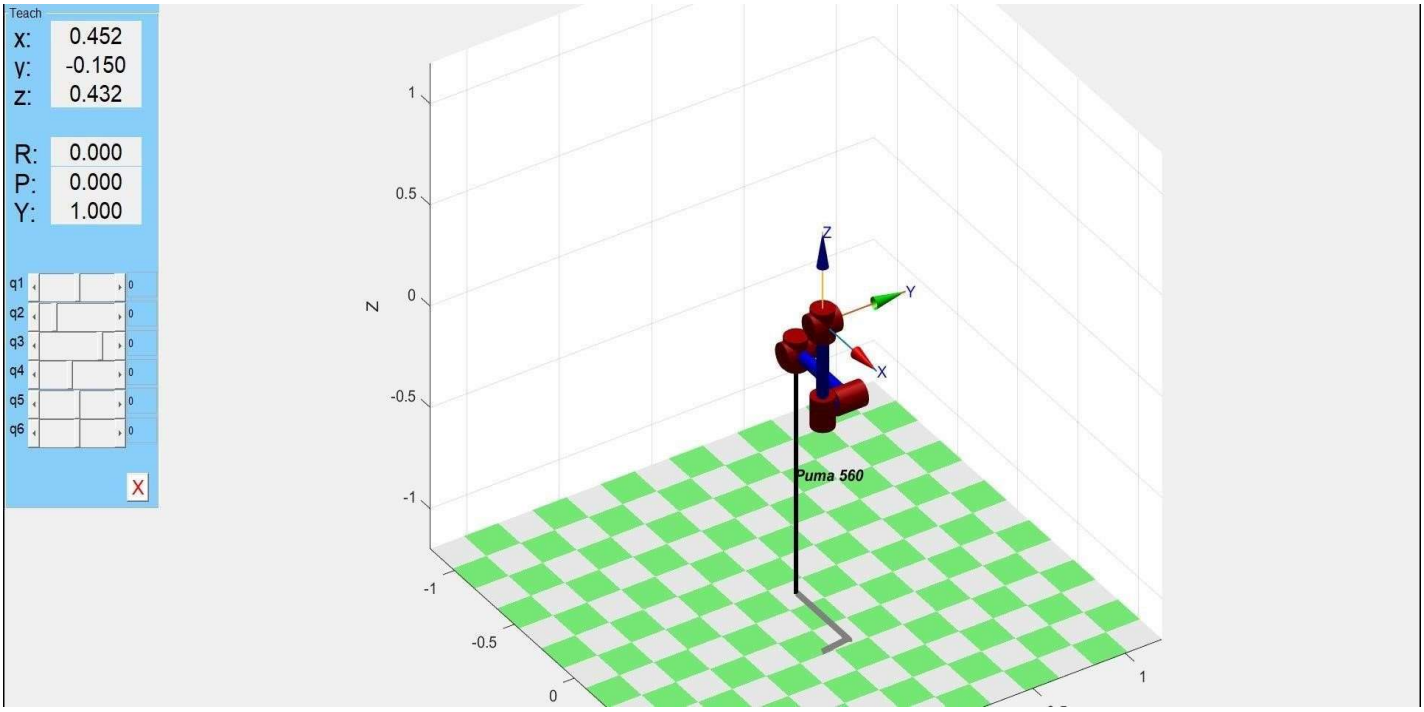
4. **Steps for experiment/practical:**

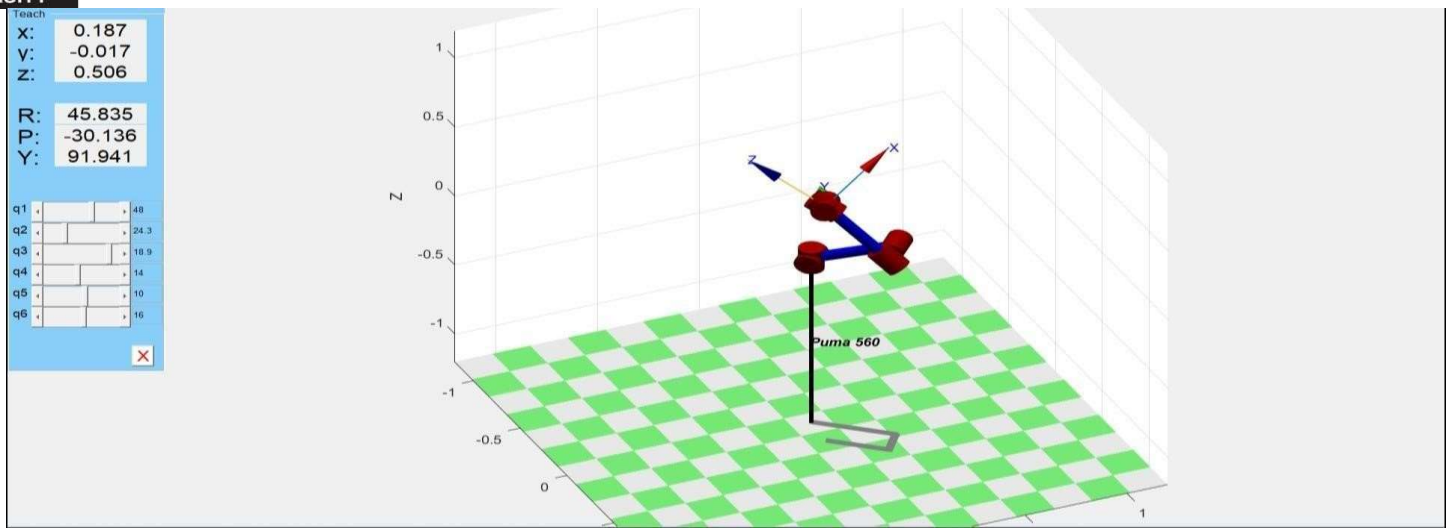
Step 1. Open MATLAB
Step 2. Open new M-file
Step 3. Copy the code given below.
Step 4. Save in current directory
Step 5. Compile and run the program **Step 6.** For output see the output window.

5. Code:-

```
Link() l(1)=Link([0,0,0,pi/2,0]);
l(2)=Link([0,0,0.4318,0,0]);
l(3)=Link([0,0.15,0.0203,-pi/2]);
l(4)=Link([0,0.4318,0,pi/2,0]);
l(5)=Link([0,0,0,-pi/2,0]);
l(6)=Link([0,0,0,0,0]);
puma=SerialLink(l)
puma.plot([0,0,0,0,0,0])
mdl_puma560 p560.teach()
```

Simulation Output:-





Results and Discussion: -

In this experiment, we learned how to design and simulate the 6 DOF robotic arm in MATLAB. Constructed with individual coding of joints and parameters, the arm's configuration was visualized using the 'plot' function. Real-time parameter manipulation during execution was facilitated by the 'teach' command, offering an interactive interface for dynamic control and analysis of the robotic arm's behavior. **Learning Outcome:-**

- Learnt about how to design and simulate 6-DOF robotic arm.
- Learnt how to generate the robotic arm by using coding through MATLAB only.
- Learnt about the **teach** command to manipulate the robotic arm according to the need.

Evaluation Grid:-

Performance (12 Marks)	**Worksheet (10 Marks)**	**Viva (8 Marks)**	**Total**
		Teacher's Sign	

University Institute of Engineering

Department of Electronics & Communication Engineering

Experiment No. :- 9

Student Name: Priyanshu Mathur **UID:** 20BEC1073

Branch: Electronics and Communication **Section/Group:** A

Semester: 7th **Date of Performance:** 15/11/2023

Subject Name: Industrial Automation & Robotics **Subject Code:** 20ECA-446

1. **Aim of the practical:** Simulate the trajectory of puma560 in joint space motion using MATLAB.

2. **Tool Used:** MATLAB

3. **Theory:**

In robotics, achieving smooth end-effector movement between two positions involves trajectory planning. MATLAB's 'mtraj' and 'jtraj' functions are pivotal for this task. 'mtraj' creates a multi-axis trajectory using a scalar function, while 'jtraj' focuses on joint space trajectory using quantic polynomials. 'tpoly' generates a scalar polynomial trajectory, and 'lspb' produces a trajectory with linear segments and parabolic blends. These functions provide flexibility for designing and executing robotic trajectories, allowing customization of velocity and acceleration outputs.

4. **Steps for experiment/practical:**

Step 1. Open MATLAB
Step 2. Open new M-file
Step 3. Copy the code given below
Step 4. Save in current directory **Step 5.** Compile and run the program **Step 6.** For output see the output window.

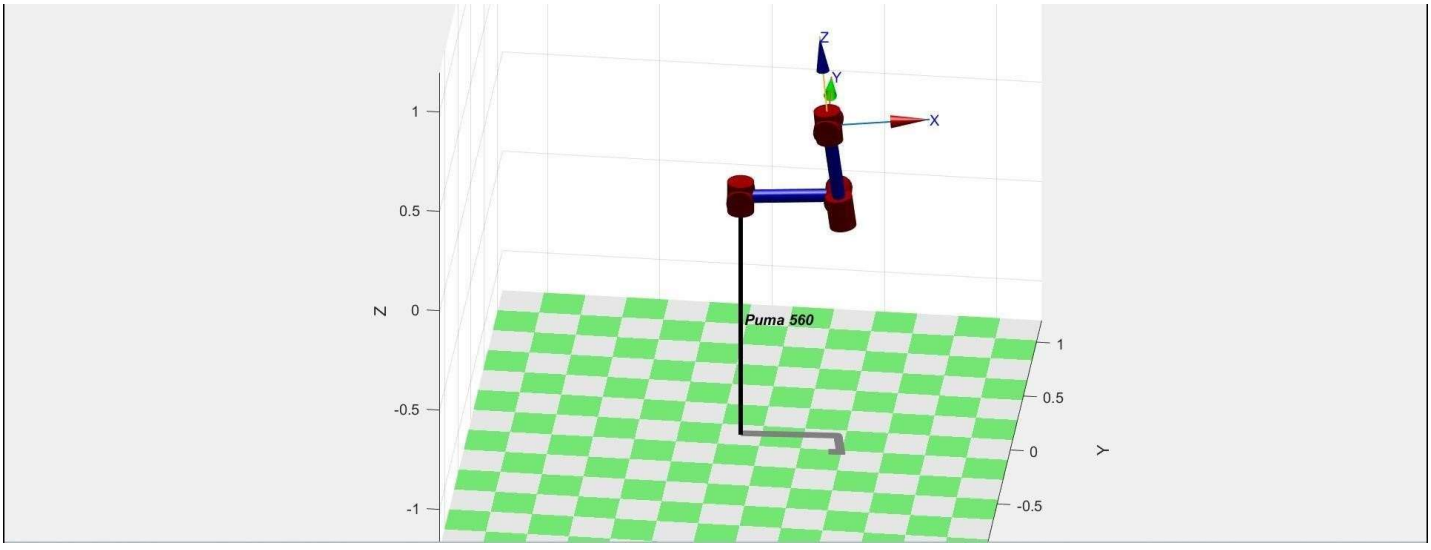
5. **Code:-**

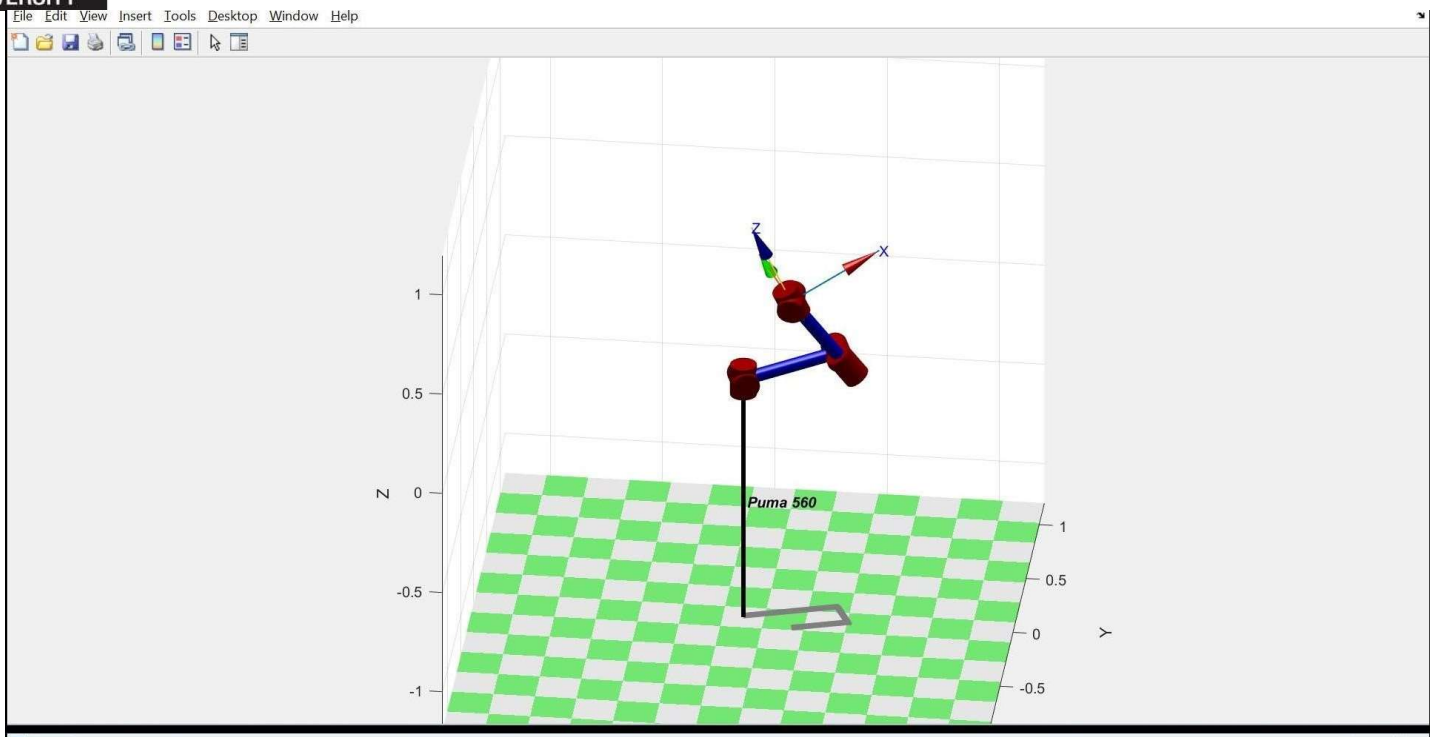
```
mdl_puma560
% Define the initial and final joint angles q0 = [0 0
0 0 0 0]; % initial joint angles (in radians) qf =
```

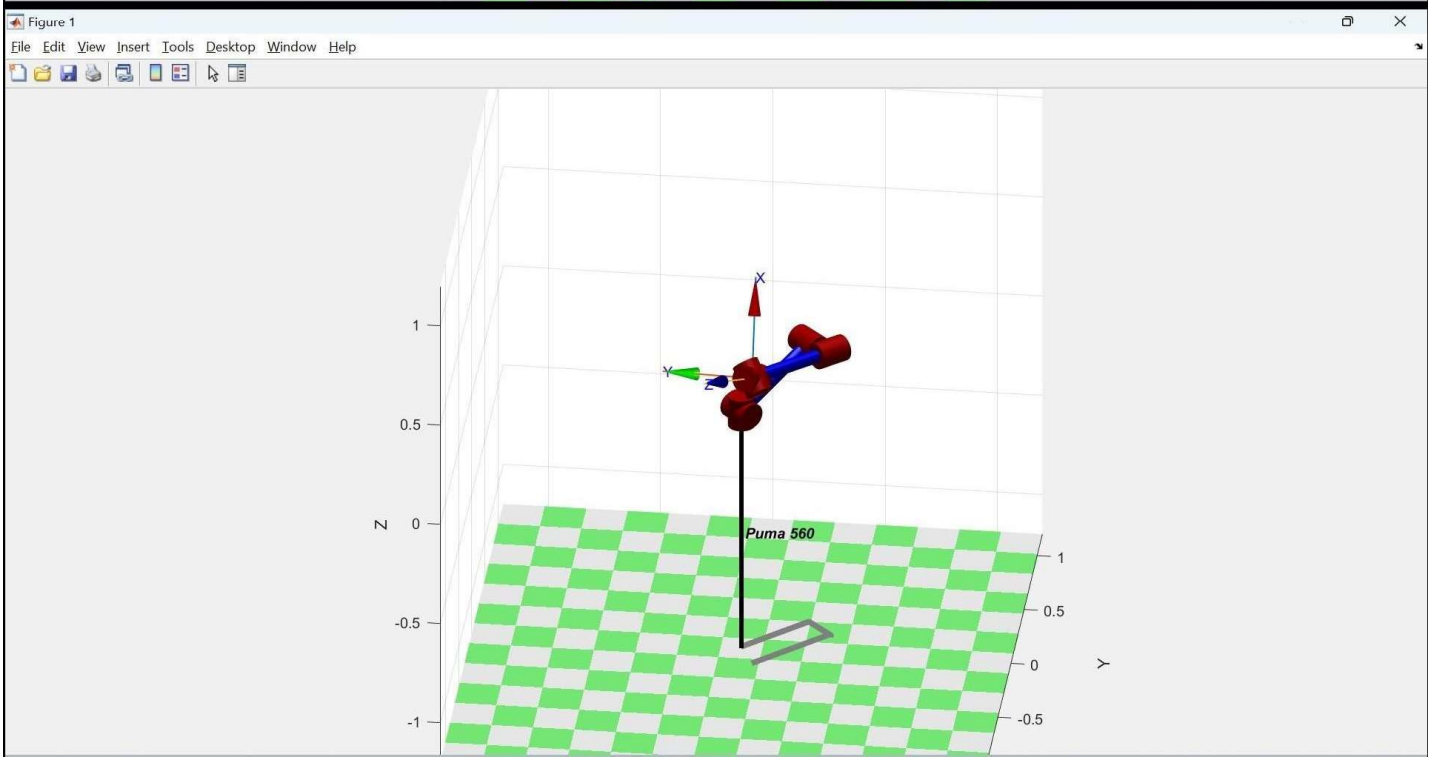
```
[pi/4 pi/6 pi/3 -pi/4 -pi/6 pi/3]; % final joint
angles (in radians)

% Define the time duration and time step
tf = 5; % time duration (in seconds) dt =
0.01; % time step (in seconds)
% Compute the number of steps
N = ceil(tf/dt);
% Compute the joint angles at each step using linear interpolation
qs = zeros(N, 6); for i = 1:N
    t = i*dt/tf; qs(i, : ) = (1-
    t)*q0 + t*qf;
end
% Simulate the robot motion
figure;  p560.plot(q0);  hold
on;     for     i     =     1:N
p560.plot(qs(i,: )); pause(dt);
end
```

Simulation Output:-







Results and Discussion: -

In this experiment, we employed 'jtraj' for joint space trajectory and linear interpolation in Cartesian space to create smooth trajectories for a PUMA 560 robotic arm. Visualization in the Robot Toolbox illustrated controlled motion between initial and final poses. 'jtraj' ensured precision with quantic polynomial trajectories, while linear interpolation offered a simpler alternative. The experiment emphasizes the vital role of trajectory planning in achieving accurate and smooth robotic movements.

Learning Outcome:-

- Learnt about the joint space trajectory generation with `jtraj.
- Learnt about the functionality of `lspb` for creating scalar trajectories with constant velocity segments and parabolic blends.
- Learnt about the `tpoly` for generating scalar polynomial trajectories, facilitating smooth transitions in multi-step trajectories.

Evaluation Grid:-

Performance (12 Marks)	**Worksheet (10 Marks)**	**Viva (8 Marks)**	**Total**
		Teacher's Sign	

University Institute of Engineering

Department of Electronics & Communication Engineering

Experiment No. :- 10

Student Name: Priyanshu Mathur **UID:** 20BEC1073

Branch: Electronics and Communication **Section/Group:** A

Semester: 7th **Date of Performance:** 15/11/2023

Subject Name: Industrial Automation & Robotics **Subject Code:** 20ECA-446

1. Aim of the practical: Create a trajectory to trace letter "A" and also find out the inverse kinematics for the same using MATLAB.

2. Tool Used: MATLAB

3. Theory:

'mstraj' represents Multi-segment multi-axis trajectory in MATLAB. It generates a trajectory matrix (KxN) for simultaneous motion of N axes across M segments. Each segment involves linear motion, connected by polynomial blends. The trajectory initiates at the initial position q0 (1xN), passes through M-1 via points defined by matrix p (MxN), and concludes at the last row of p. The matrix has one row per time step and one column per axis, with the number of steps K determined by via points and time/velocity limits.

'homtran' is employed to apply a homogeneous transformation. It transforms points stored in the column-wise matrix p using the specified transformation T. If T1 represents a 3D transformation, applying T to T1 results in tp = T * T1. In cases where T1 is an NxNxP matrix and T is an NxN matrix, the result is an NxNxP matrix, applying T to each plane defined by the first two dimensions.

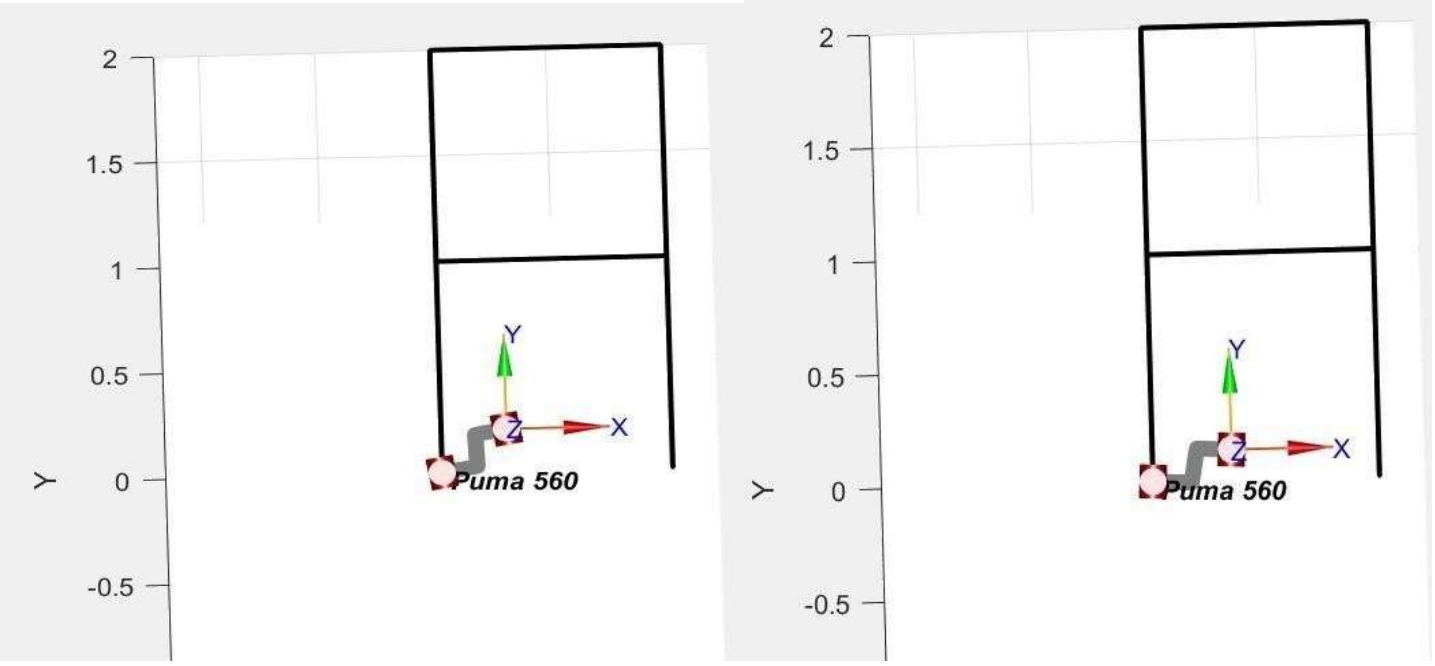
4. Steps for experiment/practical:

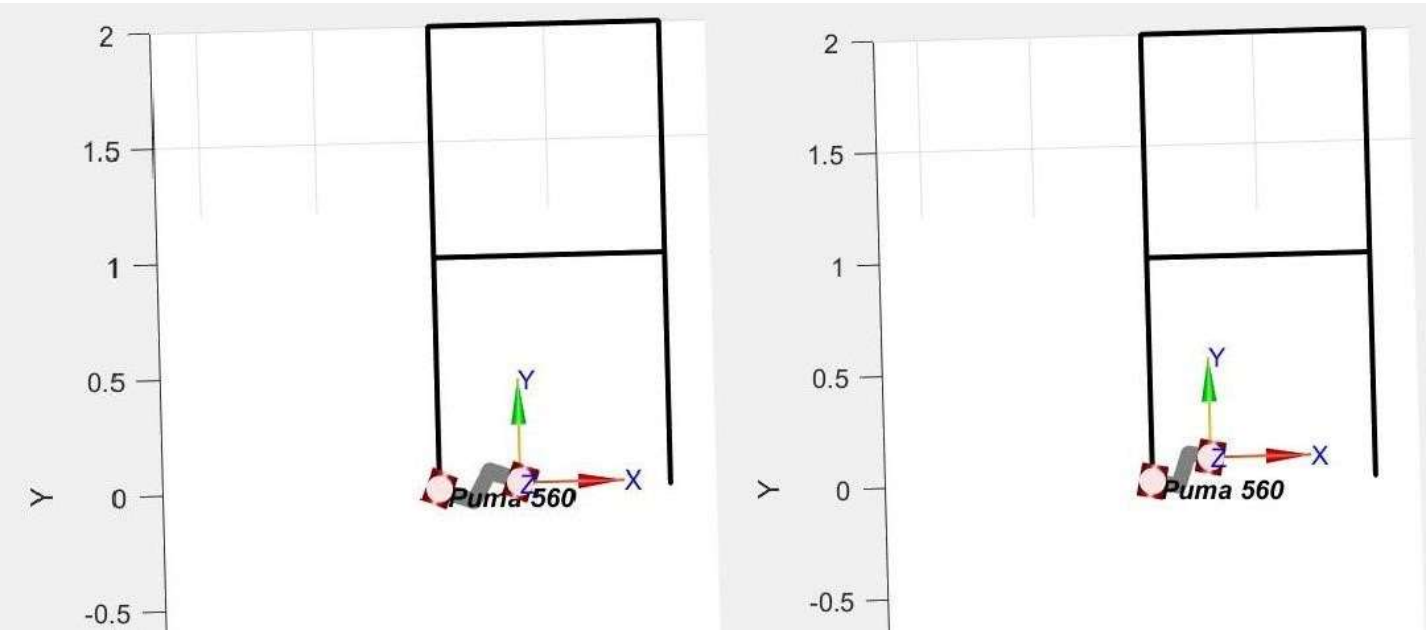
Step 1. Open MATLAB
Step 2. Open new M-file
Step 3. Copy the given code below
Step 4. Save in current directory
Step 5. Compile and run the program
Step 6. For the output see the output window.

5. Code:- mdl_puma560

```
%define a number of via points that define the strokes of the letter path =[0 0
0; 0 2 0; 1 2 0; 1 0 0; 1 1 0; 0 1 0];
figure;          %to generate the figure
%to plot the path segments plot3(path(:,1),path(:,2),path(:,3),
'color','k','LineWidth',2) %for
continuous path
p =mstraj (path,[2.5 2.5 2.5], [],[2 2 2], 0.02, 0.2);
%sequence of homogeneous transformations describing the pose at every point along the path. Tp =transl (0.1*
p);
%origin of the letter will be placed at (0.4,0,0) in the work place Tp
=homtrans( transl(0.25, 0, 0), Tp);
%orientation p560.tool=trotx
(pi);
% to determine the joint coordinates and then animate it. q
=p560.ikine6s (Tp);
%to generate the figure figure; hold
on;
%to plot puma560 p560.plot
(q)
```

Simulation Output:-


2
1.5
1
0.5
0
-0.5
Y
Y
X
Z
Puma 560


2
1.5
1
0.5
0
-0.5
Y
Y
X
Z
Puma 560

Results and Discussion: -

In this experiment, we created a trajectory for the PUMA 560 robotic arm to trace a letter's strokes defined by via points in 3D space. The path matrix specified via points, and the mstraj function generated a continuous trajectory with set velocity limits. Homogeneous transformations described the pose at each point. Inverse kinematics computed joint coordinates, enabling animation of the PUMA 560 to trace the letter.

Learning Outcome:-

- Learnt how to generate robotic arm trajectories using 'mstraj' function
- Learn to apply homogeneous transformations to define the trajectory of robotic arm..
- Learnt how to use joint coordinated for tracing the letter.

Evaluation Grid :-

Performance (12 Marks)	**Worksheet (10 Marks)**	**Viva (8 Marks)**	**Total**
		Teacher's Sign	