Q3.1:

Ans: Following are the results for the implemention of an impedance controller and a force controller such that the arm generates 15 N of force against the static whiteboard:

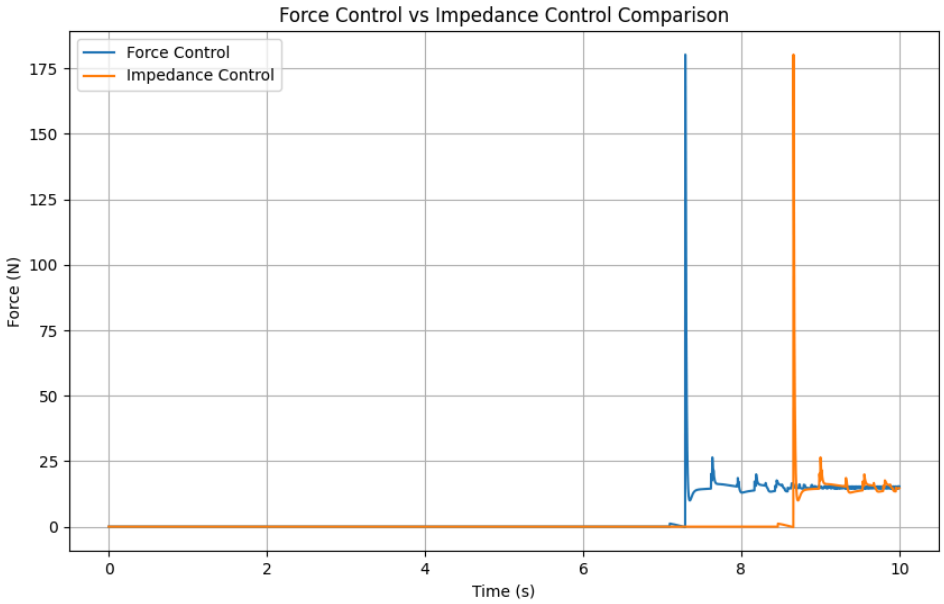

Figure 3.1.1: Juxtaposed graphs of Force vs time for force controller and impedance controller

The graph depicts the force applied over time by two different types of controllers: a force controller and an impedance controller. The behavior of the force controller shows that it rapidly reaches a peak force, overshooting significantly past the desired 15 N before stabilizing close to the target force. This indicates a more aggressive control strategy that does not account for the dynamic properties of the arm and environment, which leads to a less stable initial response. On the other hand, the impedance controller graph shows a gradual increase to the desired force level with minimal overshoot, followed by a relatively stable maintenance of the force. This suggests that the impedance controller is taking into account the mechanical impedance of the arm and the interaction with the environment, which results in a more controlled and smoother approach to reaching the desired force without significant oscillations or overshoot.

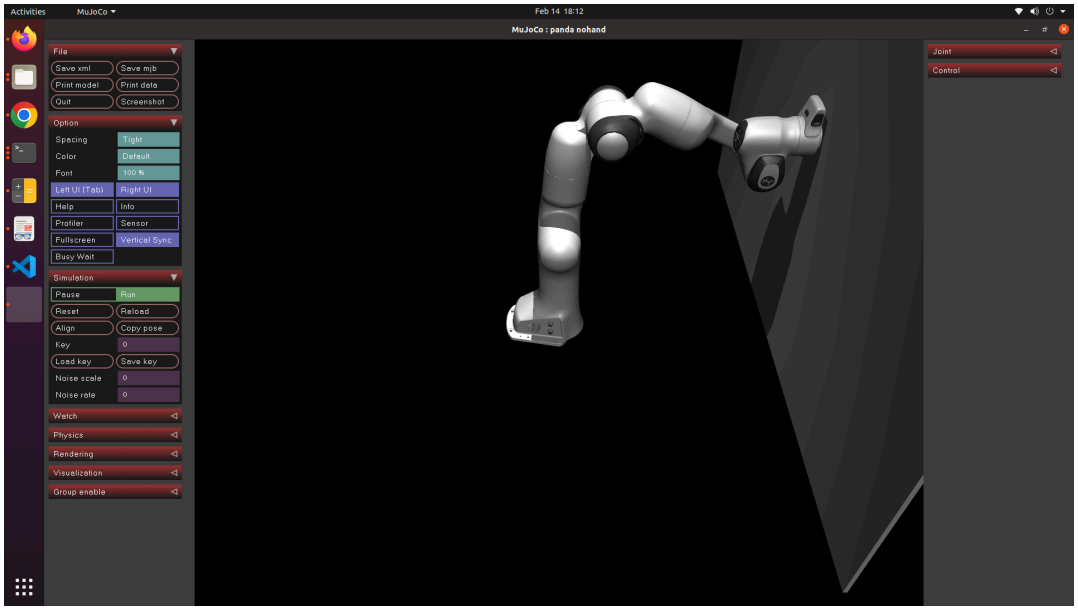

Figure 3.1.2: Robot when the board is near its maximum amplitude (Force Control)

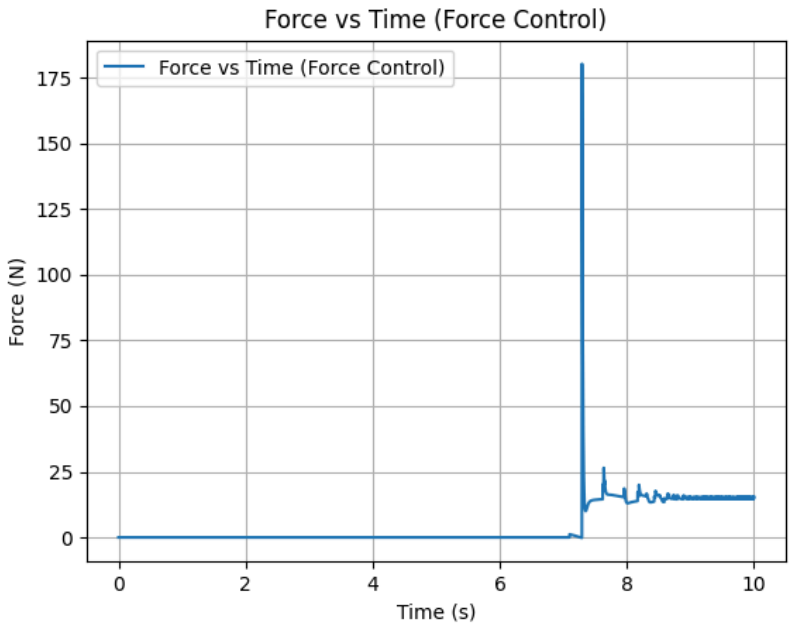

Figure 3.1.3: Graph of Force vs time for Force Controller

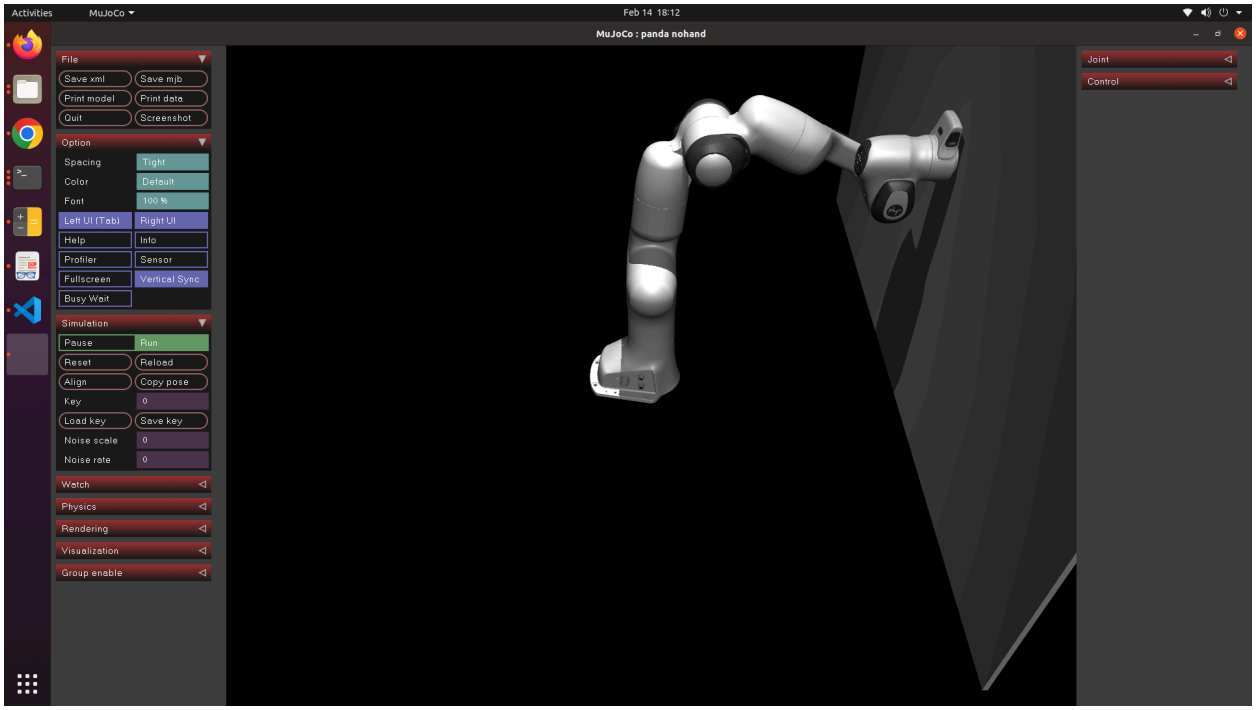

Figure 3.1.4: Robot when the board is near its maximum amplitude (Impedance Control)

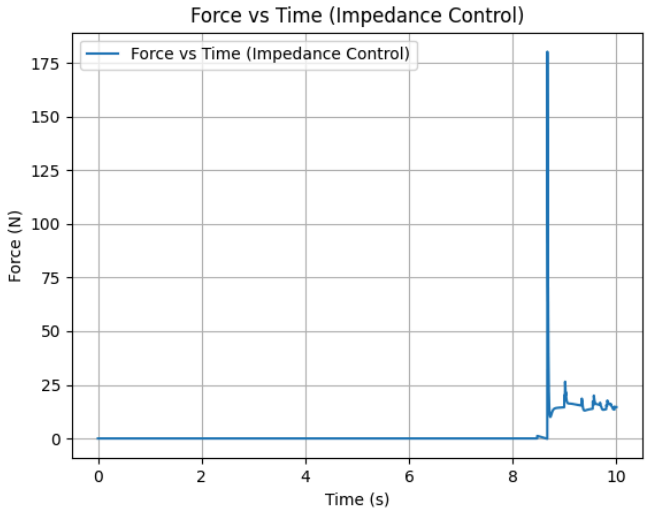

Figure 3.1.5: Graph of Force vs time for Impedance Controller

Q3.2:

Ans: Following are the results for the implemention of an impedance controller and a force controller such that the arm generates 15 N of force against the whiteboard with disturbance in the form of a sinusoidal motion:

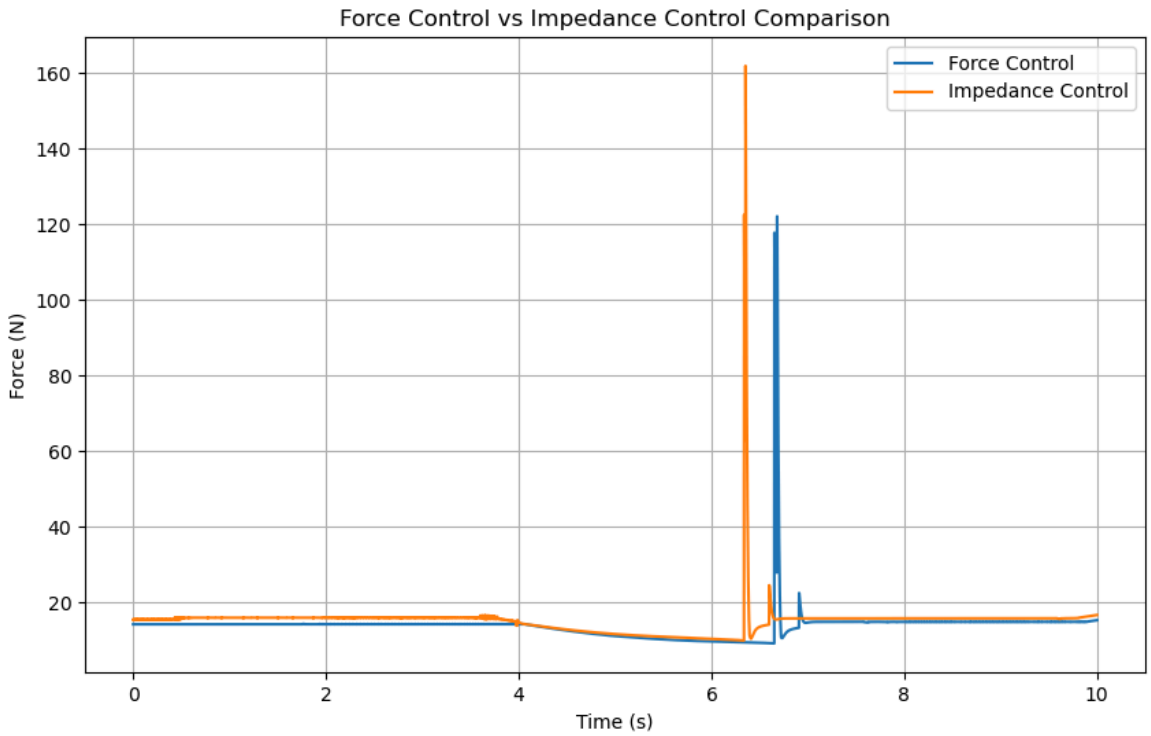

Figure 3.2.1:Juxtaposed graphs of Force vs time for force controller and impedance controller

The force control's response to the disturbance is characterized by very sharp and high-amplitude spikes, suggesting a more aggressive and less stable approach to maintaining the desired force against the moving whiteboard. It seems to react abruptly to changes, which can cause instability in dynamic environments. In contrast, the impedance control displays fewer and lower spikes, suggesting that while it also responds to the moving whiteboard, it does so in a more damped and controlled manner. This behavior is indicative of the impedance control's ability to better handle dynamic interactions with the environment by adjusting the mechanical impedance, resulting in a more robust performance against the oscillatory movement of the whiteboard.

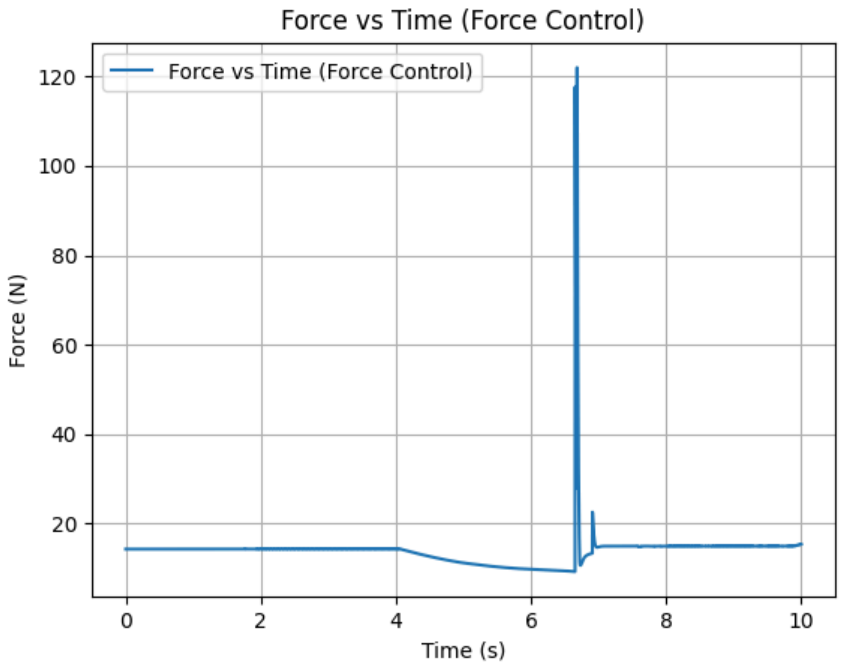

Figure 3.2.2:Graph of Force vs time for Force Controller

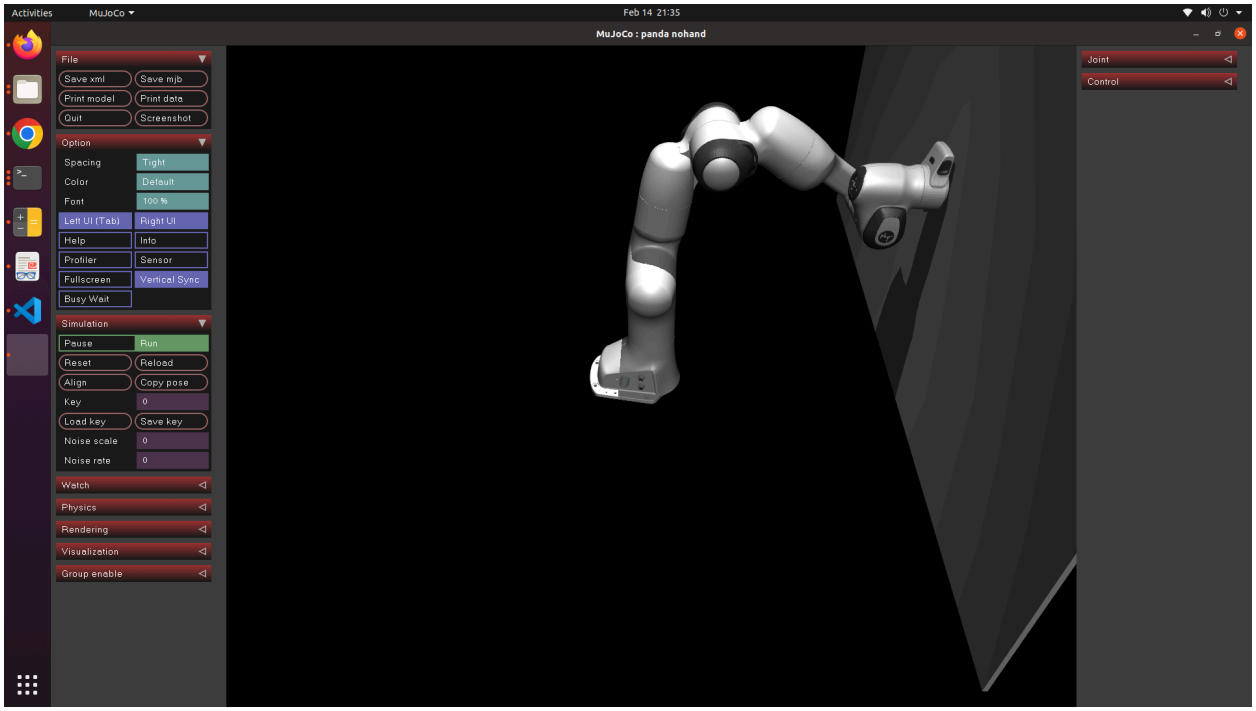

Figure 3.2.3:Robot when the board is near its maximum amplitude (Force Control)

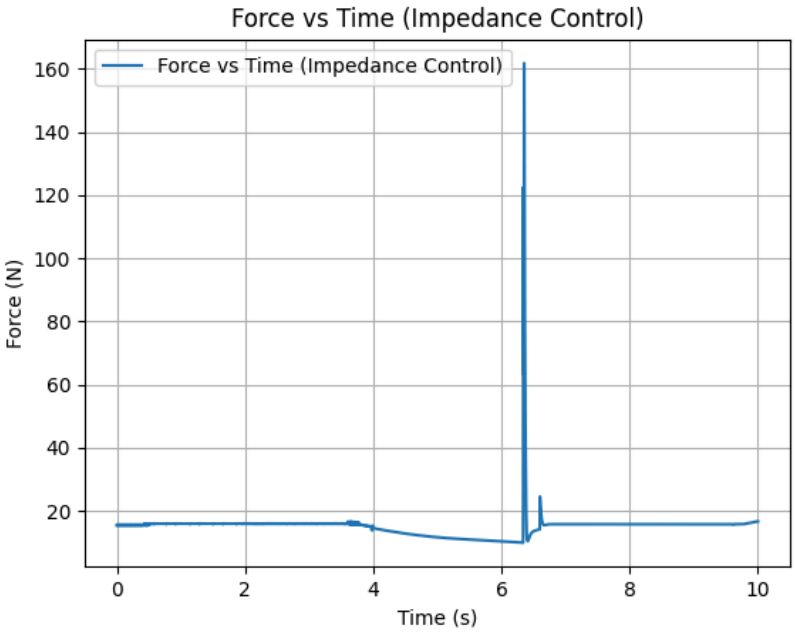

Figure 3.2.4:Graph of Force vs time for Impedance Controller

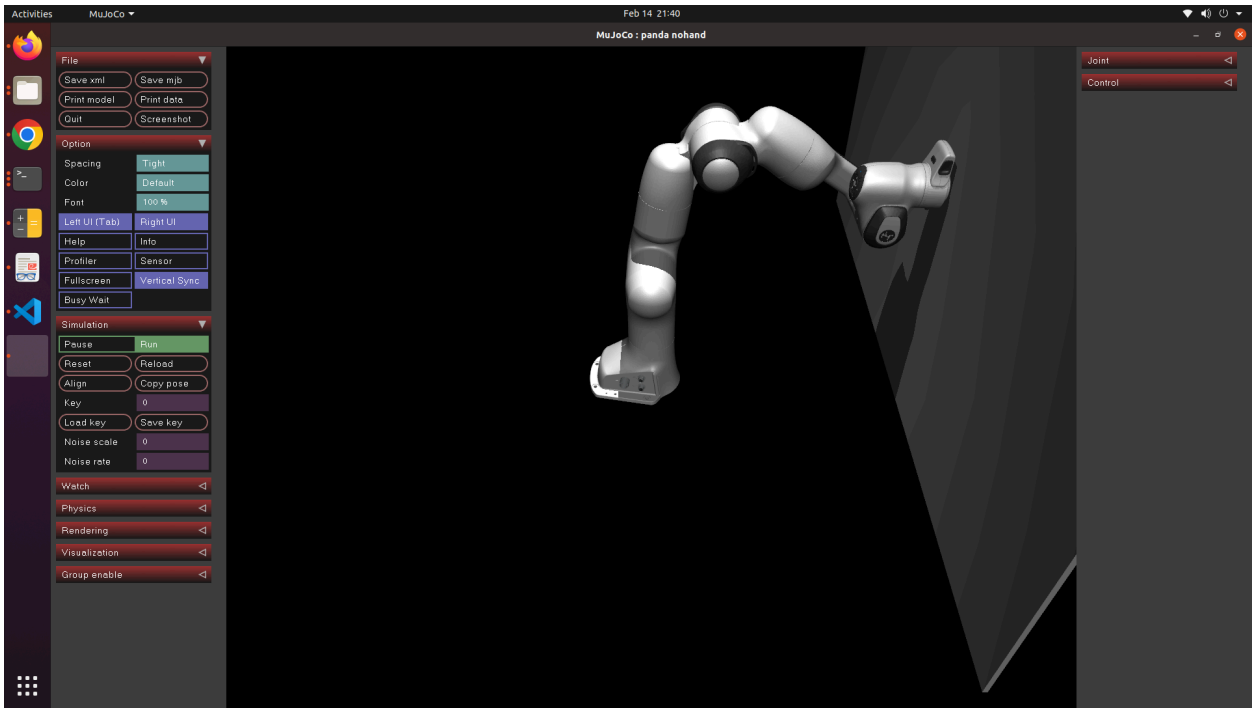

Figure 3.2.5:Robot when the board is near its maximum amplitude (Impedance Control)

Q4.1:

Ans: In this question we were asked to implement the forward kinematics and perform the computation of the Jacobian for a Franka arm by implementing code within the ForwardKin function, designed to calculate the pose (position and orientation) of each joint, as well as the end-effector, based on given joint angles, and to compute the Jacobian matrix, which represents the relationship between joint velocities and the end-effector's velocity. The process unfolds as follows:

- Initialization of Joint Angles: The function begins by updating the array of joint angles with the provided input angles. This is crucial for ensuring that the calculation reflects the current configuration of the robotic arm.

- Base Joint Transformation: For the base joint (the first joint of the robot), its transformation matrix is computed using a standard transformation function. This function takes into account the joint's rotation (around the z-axis in this case) and its position, which is initially set at the origin. This establishes the foundational pose from which subsequent transformations are calculated.

- Sequential Computation of Joint Transforms: The method iteratively calculates the transformation matrices for each joint beyond the first. It does so by:
    - Computing the individual transformation for each joint based on its angle and a fixed orientation (since the rotation is primarily around the z-axis).
    - Accumulating transformations through matrix multiplication to propagate the pose from the base towards the end-effector. This step involves multiplying the current joint's transformation by the accumulated transformation matrix of all preceding joints, effectively chaining the poses together.

- Jacobian Matrix Computation: Parallel to computing the joint transforms, the Jacobian matrix is constructed to capture how variations in joint angles influence the end-effector's velocity. This involves:
    - Identifying the rotation axis for each joint and the vector pointing from the joint to the end-effector.
    - For each joint, calculating the cross product of its rotation axis and the vector to the end-effector for the rotational components of the Jacobian, and directly using the rotation axis for the linear components. This step is fundamental in understanding the kinematic relationship between joint movements and the resulting end-effector motion.

- Output: The function ultimately returns two key outcomes:
    - A set of transformation matrices, one for each joint, which describe the pose of the joint relative to the base frame.
    - The Jacobian matrix, which is critical for analyzing and controlling the arm's dynamics and kinematics.

Following are the results:

```
Joints:
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
computed FK ee position
[ 8.80000000e-02 -8.93992163e-18  9.26000000e-01]
computed FK ee rotation
[[ 1.0000000e+00  0.0000000e+00  0.0000000e+00]
 [ 0.0000000e+00 -1.0000000e+00 -1.2246468e-16]
 [ 0.0000000e+00  1.2246468e-16 -1.0000000e+00]]
----------------------------------------
```

Figure 4.1.1: FK result for q1 = [0°, 0°, 0°, 0°, 0°, 0°, 0°]

```
Joints:
[0, 0, -0.7853981633974483, -0.2617993877991494, 0.3490658503988659, 0.2617993877991494, -1.3089969389957472]
computed FK ee position
[ 0.15710277 -0.10259332  0.93602711]
computed FK ee rotation
[[ 0.64935398  0.75871099  0.05193309]
 [ 0.7552124  -0.65137389  0.07325497]
 [ 0.08940721 -0.00834789 -0.99596017]]
----------------------------------------
```

Figure 4.1.2: FK result for q2 = [0°, 0°, -45°,-15°,20°,15°,-75°]

```
Joints:
[0, 0, 0.5235987755982988, -1.0471975511965976, -1.1344640137963142, 0.7853981633974483, 0.0]
computed FK ee position
[0.40136375 0.08742801 0.85526363]
computed FK ee rotation
[[ 0.98015816 -0.18113365 -0.08050201]
 [-0.17410263 -0.5925751  -0.78647507]
 [ 0.09475362  0.78488557 -0.61235316]]
----------------------------------------
```

Figure 4.1.3: FK result for q3 = [0°, 0°, 30°, -60°,-65°,45°,0°]

Q4.2:

Ans: In this question, the IterInvKin method was implemeted to iteratively converge upon joint angles that align the arm's end-effector with a desired pose. The methodology employed leverages the damped least squares method, incorporating predefined weight matrices for W and C to modulate the influence of respective joints and dampen the solution to avoid singularities and ensure numerical stability. The implementation is as follows:

- **Parameter Definition and Initialization:** The method begins by setting the joint angles to the starting configuration and defining the weight matrices W and C as specified, which is the basis in the damped least squares approach. The matrices are designed to prioritize certain movements and stabilize the inversion process.

- **Establishing Error Tolerance and Iteration Limits:** To guide the iterative process, tolerances for positional and rotational errors are set alongside a maximum iteration count. These parameters ensure the algorithm seeks an accurate alignment within reasonable computational bounds.

- **Decomposing the Target Pose:** The target end-effector pose, encapsulated within a transformation matrix T_{Goal}, is dissected into its rotational (R_{Goal}) and translational (T_{Goal}) components, facilitating separate handling in the alignment process.

- **Iterative Refinement Process:** Central to the implementation lies the iterative loop, within which the current end-effector pose is updated using forward kinematics based on the current estimate of the joint angles. The discrepancy between the current and desired poses, both in position and orientation, is quantified. Orientation error is refined to a manageable scale using axis-angle representation and clipping techniques to ensure it remains within predefined limits.

- **Error Projection and Joint Angle Adjustment:** The core of the damped least squares approach manifests in projecting the pose error into the joint space using a pseudo-inverse of the Jacobian matrix, tempered by the W and C matrices. This projection computes the incremental adjustments needed for the joint angles. The joint angles are updated accordingly, incrementally steering the end-effector towards the target pose.

- **Convergence Evaluation and Output:** The iterative process is subjected to continual evaluation against the convergence criteria, based on error magnitudes and iteration count, ensuring the algorithm halts upon achieving acceptable alignment or exhausting the iteration limit. Upon concluding the iterations, the method yields the computed joint angles that approximate the desired end-effector pose, alongside the final error vector, indicating the residual discrepancy between the achieved pose and the target.

Following are the results:

```
----------------------------------------
Converged in 90 iterations
Error 1.0664462979142164e-06 0.0006627689536126888
Computed IK angles [0.6792348249355901, 0.3438334231337678, -0.814494493823651
9, -2.0759894700328583, -0.14629918318244606, 3.919081275572654, -2.7875448645
113576]
(16662_env) shahram95@shahram95:~/Desktop/Homework folder/16662_S24/16_662_HW1
/Kinematics$ code .
```

Figure 4.2.1: Final joint angles for moving to the end effector goal pose of R_g , t_g

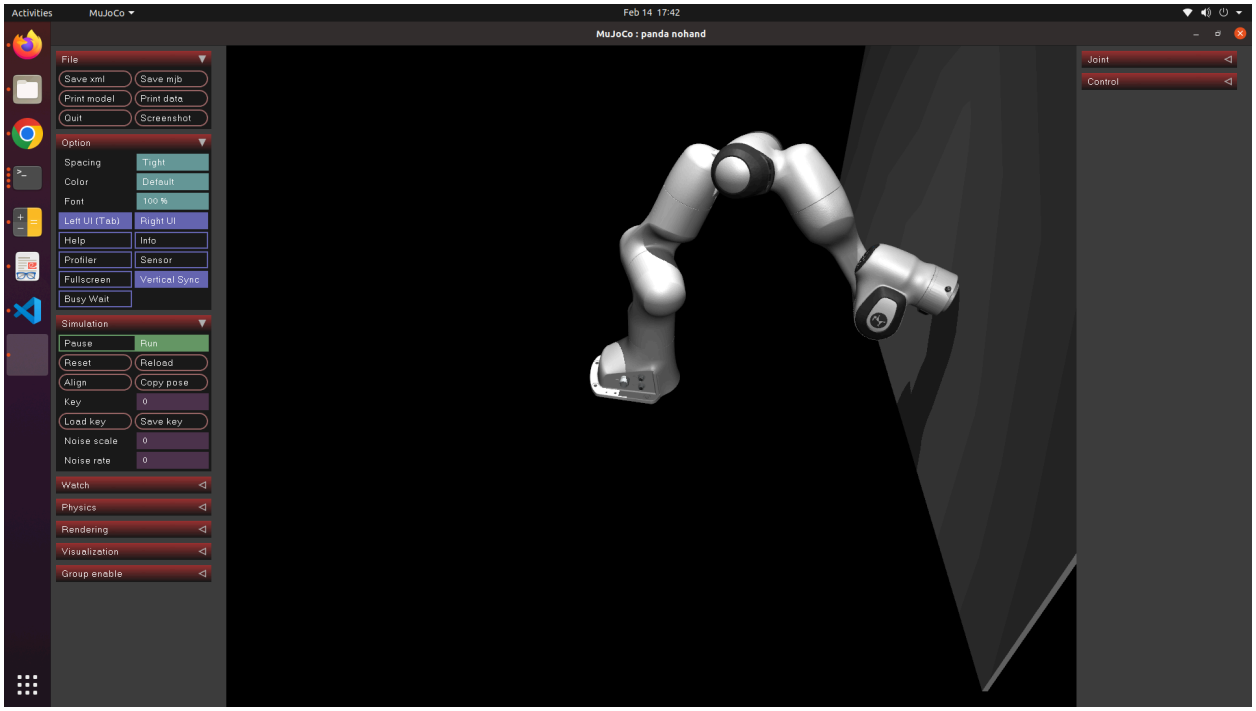

Figure 4.2.2: Franka robot to the joint positions from the IK