Yinhong Qin
yq2021@nyu.edu

Project 2

ROB 6003
December 21, 2021

Question 1

1. We will use the iterative numerical method to solve the inverse kinematics problem.

   Given a joint angle θ, the pose of end-effector T could be calculated with forward kinematics algorithms:

   $$T = forward\_kinematics(\theta) = fk(\theta)$$

   The inverse kinematics solution is to find a joint angle θ for a given pose T, which is exactly to solve the equation above.

   The taylor expansion of $fk(\theta)$ is:

   $$fk(\theta) = fk(\theta_0) + \frac{\partial fk}{\partial \theta}(\theta_0)(\theta - \theta_0) + h.o.t$$

   Use the first two terms and solve the equation

   $$fk(\theta) = x_{desired}$$

   (For Question 1, we only consider the translation part of the end-effector pose.)

   We can have

   $$x_{desired} - fk(\theta) = J\Delta\theta$$

   Here, $J_{3\times 7}$ is the last 3 rows of the Jacobian at the frame that is located at the end-effector frame origin but oriented as the spatial frame.

   Then we can have

   $$\Delta\theta = J^{-1}(x_{desired} - fk(\theta))$$

   Since J is not square, J^{-1} is the Moore–Penrose pseudo inverse. Then the iterative step is:

   $$\theta_{i+1} = \theta_i + \alpha * \Delta\theta$$

   The convergence condition is

   $$\|x_{desired} - fk(\theta)\| \leq \epsilon$$

   Here α and ϵ are both hyper parameters that can be adjusted to get an accurate result.

   To avoid the singularity problem when we computing the pseudo inverse of the jacobian, here we perform the Tikhonov regularization method. It uses this calculation listed below to replace the pseudo inverse:

   $$J^{\dagger} \approx J^T(JJ^T + \lambda I)^{-1}$$

   where λ is a very small value, $\lambda \in [10^{-4}, 10^{-8}]$.

For these 10 given goal positions, the error of our inverse kinematics solution is listed as below.

goal 1	goal 2	goal 3	goal 4	goal 5
0.28121	9.695×10^{-6}	9.857×10^{-6}	9.481×10^{-6}	9.261×10^{-6}

goal 6	goal 7	goal 8	goal 9	goal 10
0.0658	0.36063	0.04358	9.315×10^{-6}	9.359×10^{-6}

From the results, we can notice that the robot can reach the goal 2,3,4,5,9,10. Below is the visualization of solution to goal position 2.

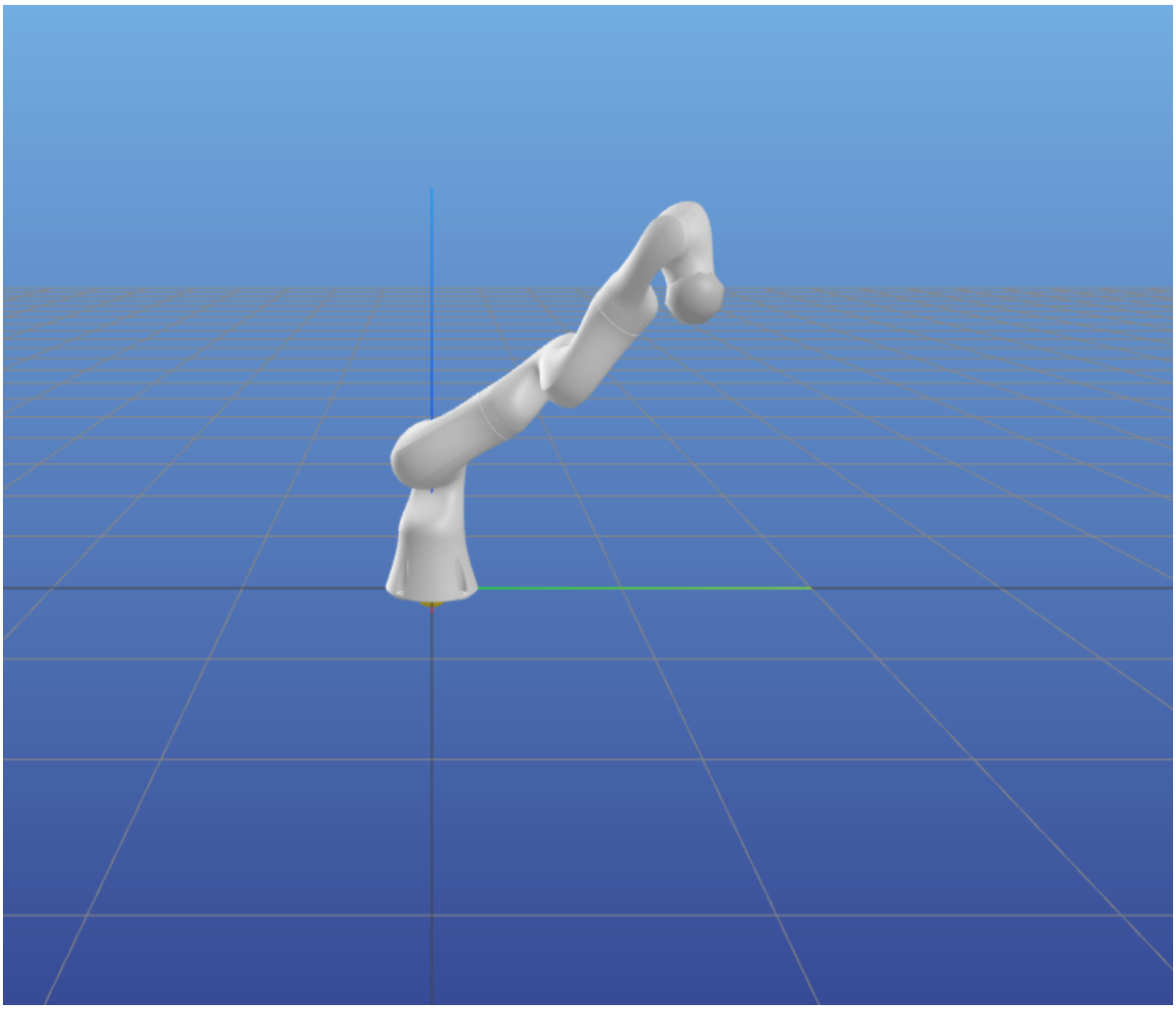

Figure 1: Inverse kinematics solution for goal position 2

For goal position 1,6,7,8, the robot cannot reach them precisely because they may fall out of the configuration space of the robot. Below is the visualization of solution to goal position 1.

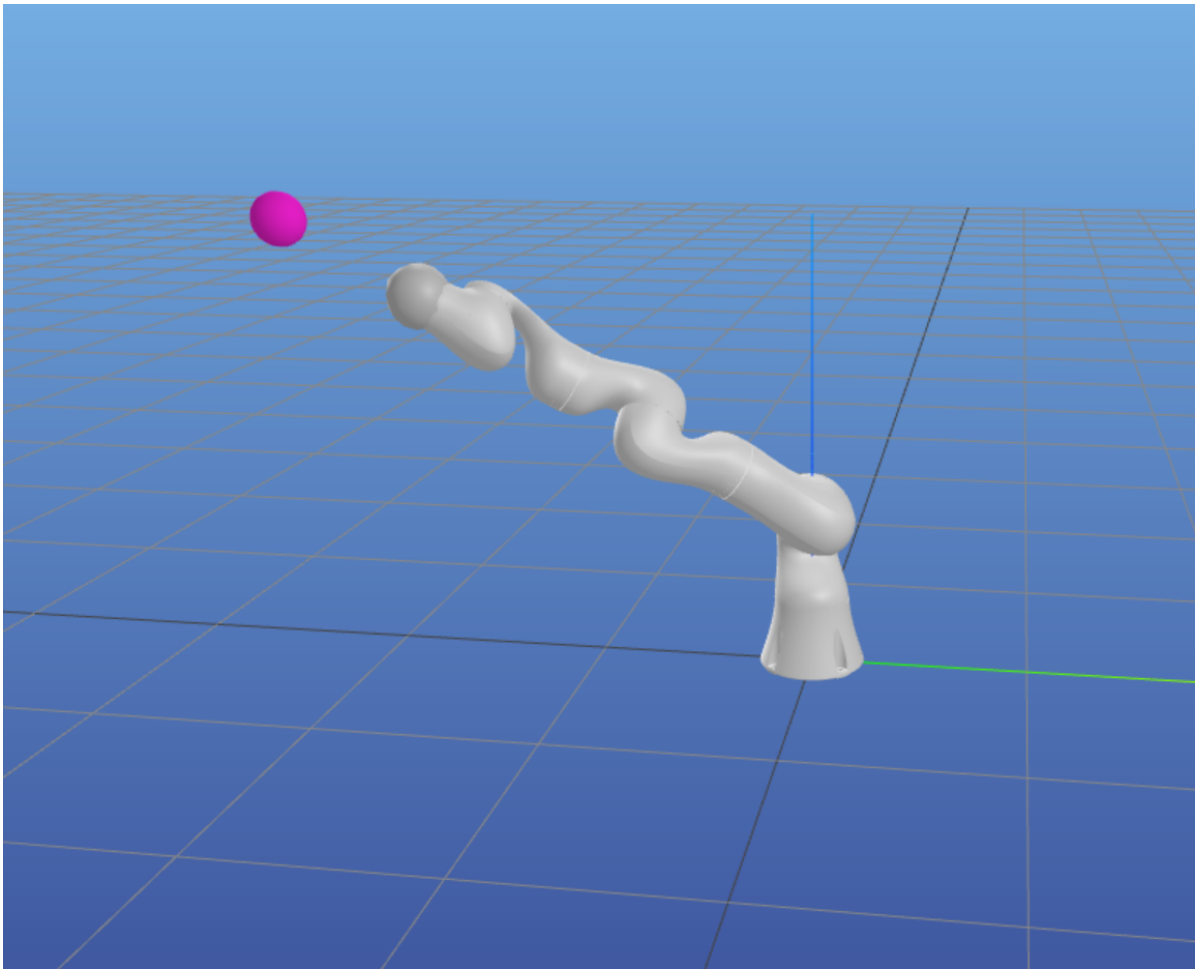

Figure 2: Inverse kinematics solution for goal position 1

2. The nullspace projector can utilize the redundancy of the 7 degree of freedom robot. It will try and keep the robot close to a given joint configuration $\bar{\theta}$.

   It is exactly the answer to this optimization problem:

$$\min_x \frac{1}{2}\|x - x_0\|^2$$
$$s.t.\ Jx = 0$$

   Solving this problem by Lagrangian multiplier, we can have

$$x = (I - J^\dagger J)x_0$$

It can be solved from another point of view. Since the psuedo-inverse of the special Jacobian of this question always satifies:

$$JJ^{\dagger}J = J$$

Then for an arbitrary joint angle θ:

$$JJ^{\dagger}J\theta = J\theta$$
$$J(I - J^{\dagger}J)\theta = 0$$

The second line above also indicates that vector $(I - J^{\dagger}J)\theta$ is in the nullspace of J.

Therefore, our error term $\Delta\theta$ now become:

$$\Delta\theta = J^{-1}(x_{desired} - fk(\theta)) + (I - J^{\dagger}J)\bar{\theta}$$

where $\bar{\theta}$ is the difference between a given joint angle and current joint angle.

The first part of $J^{-1}(x_{desired} - fk(\theta))$ drives the robot to be closer to desired position and the second part $(I - J^{\dagger}J)\bar{\theta}$ drives the robot to be as close as to a given configuration.

With the nullspace projector, the updated results of 10 given goals are:

goal 1	goal 2	goal 3	goal 4	goal 5
0.28121	9.702×10^{-6}	4.095×10^{-5}	4.768×10^{-5}	9.277×10^{-6}

goal 6	goal 7	goal 8	goal 9	goal 10
0.0669	0.3601	0.0425	6.0415×10^{-5}	2.208×10^{-5}

The visualization results of solution with nullspace projector of goal 1, 2 are listed below:

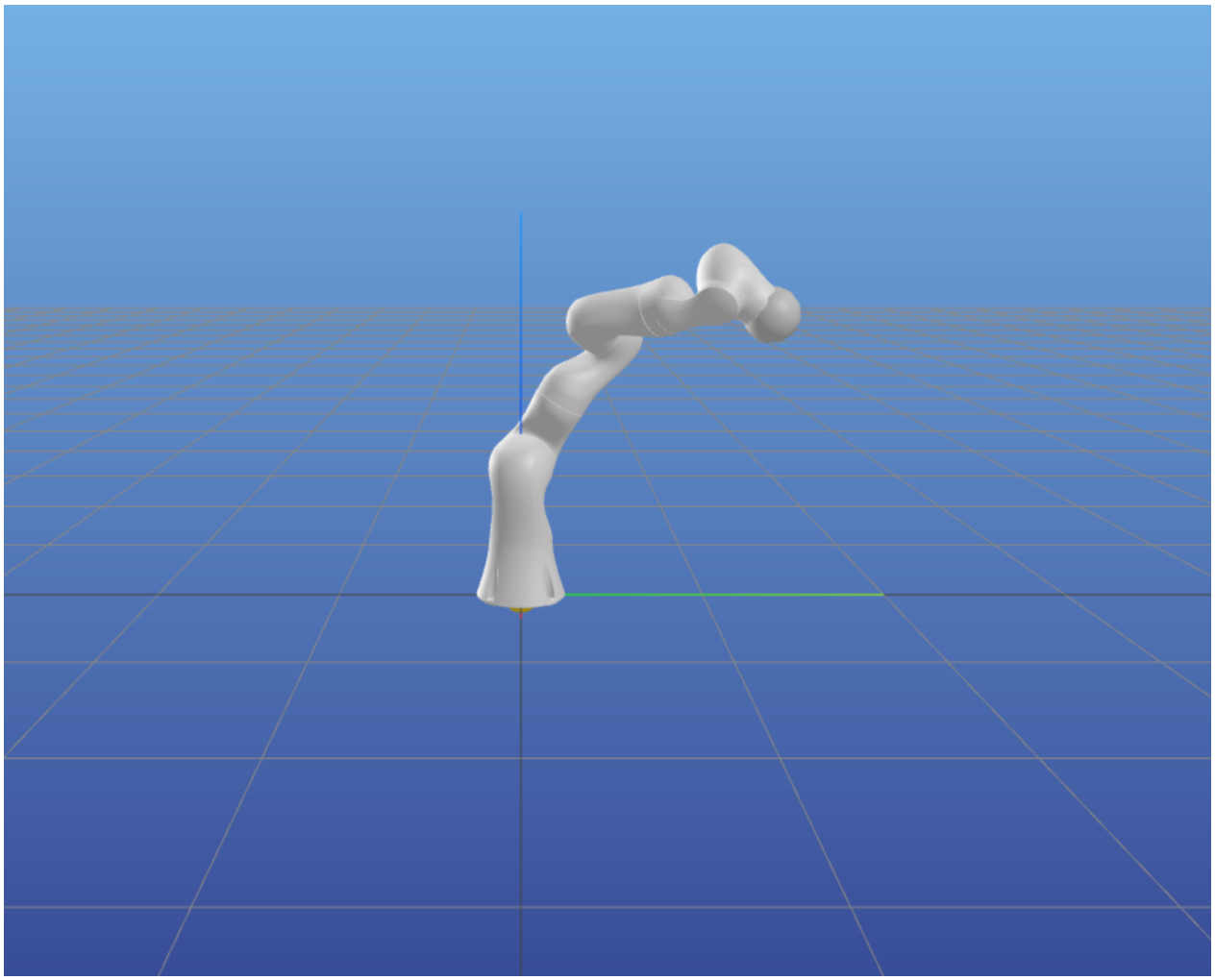

Figure 3: Inverse kinematics solution with nullspace projector for goal position 2

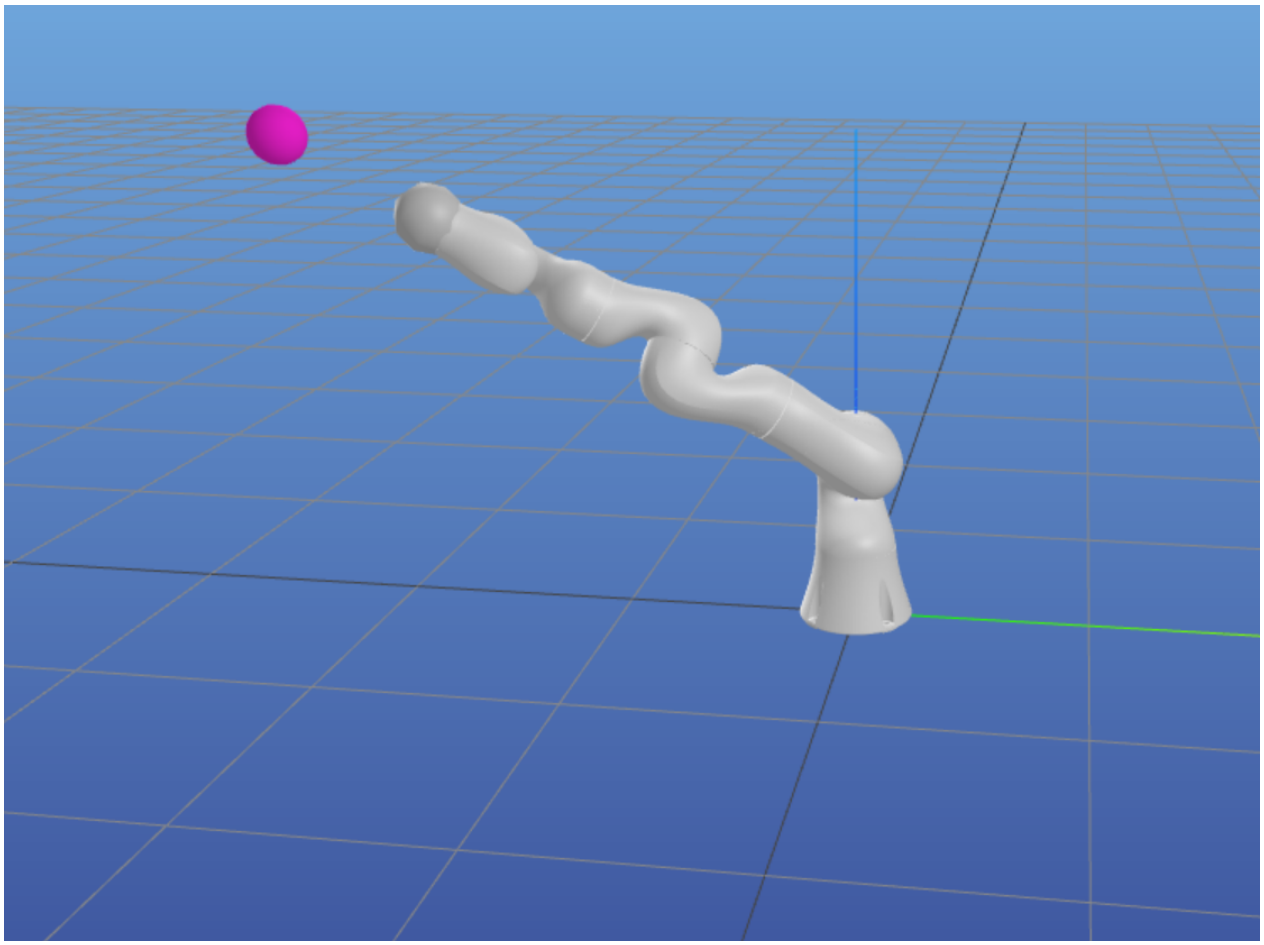

Figure 4: Inverse kinematics solution with nullspace projector for goal position 1

3. We can notice that, for a same goal position, the answer derived from differecnt methods are close to each other. But the pose of the robot are different since the nullspace projector will drive the robot to get as closer as to a given configuration.

Question 2

1. For each goal position $x_{goal} \in \mathbb{R}^3$, use the solution inverse kinematics mentioned in Question 1 to compute the related joint angles $\theta_{goal} \in \mathbb{R}^7$.

   Assume we start with joint angle θ_{start} and should reach θ_{goal} at $t = T$. To generate a trajectory in order to ensure both the velocity and acceleration are 0 at $t = 0$ and $t = T$, we can use the 5-order polynomial interpolation method.

   The joint angle at each time t can be written in such function:

   $$\theta(t) = \theta_{start} + P(t)(\theta_{goal} - \theta_{start})$$

   where $P(t)$ is a 5-order polynomial of t:

   $$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

   To ensure both the velocity and acceleration are 0 at $t = 0$ and $t = T$, we can extract these equations below

   $$\begin{cases} \theta(t=0) = \theta_{start} \\ \theta(t=T) = \theta_{goal} \\ \dot{\theta}(t=0) = 0 \\ \dot{\theta}(t=T) = 0 \\ \ddot{\theta}(t=0) = 0 \\ \ddot{\theta}(t=T) = 0 \end{cases}$$

   which is exactly

   $$\begin{cases} \theta_{start} + a_0(\theta_{goal} - \theta_{start}) = \theta_{start} \\ \theta_{start} + (a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 + a_5 T^5)(\theta_{goal} - \theta_{start}) = \theta_{goal} \\ a_1(\theta_{goal} - \theta_{start}) = 0 \\ (a_1 + 2a_2 T + 3a_3 T^2 + 4a_4 T^3 + 5a_5 T^4)(\theta_{goal} - \theta_{start}) = 0 \\ a_2(\theta_{goal} - \theta_{start}) = 0 \\ (2a_2 + 6a_3 T + 12a_4 T^2 + 20a_5 T^3)(\theta_{goal} - \theta_{start}) = 0 \end{cases}$$

   We can know the coefficients of this polynomial by solving the linear equations above:

   $$\begin{cases} a_0 = 0 \\ a_1 = 0 \\ a_2 = 0 \\ a_3 = \frac{10}{T^3} \\ a_4 = -\frac{15}{T^4} \\ a_5 = \frac{6}{T^5} \end{cases}$$

   Therefore, the trajectory generation function is:

   $$\theta(t) = \theta_{start} + (\frac{10}{T^3}t^3 + -\frac{15}{T^4}t^4 + \frac{6}{T^5}t^5)(\theta_{goal} - \theta_{start})$$

An additional detail for question 2 is to normalize the inverse kinematics solution to ensure every joint angle lies in the range $[-\pi, \pi]$. This can prevent large input torque for the robot.

2. The desired and actual joint angle positions and velocities are plotted below. The yellow line represents the actual variable and the blue line represents the desired variable.

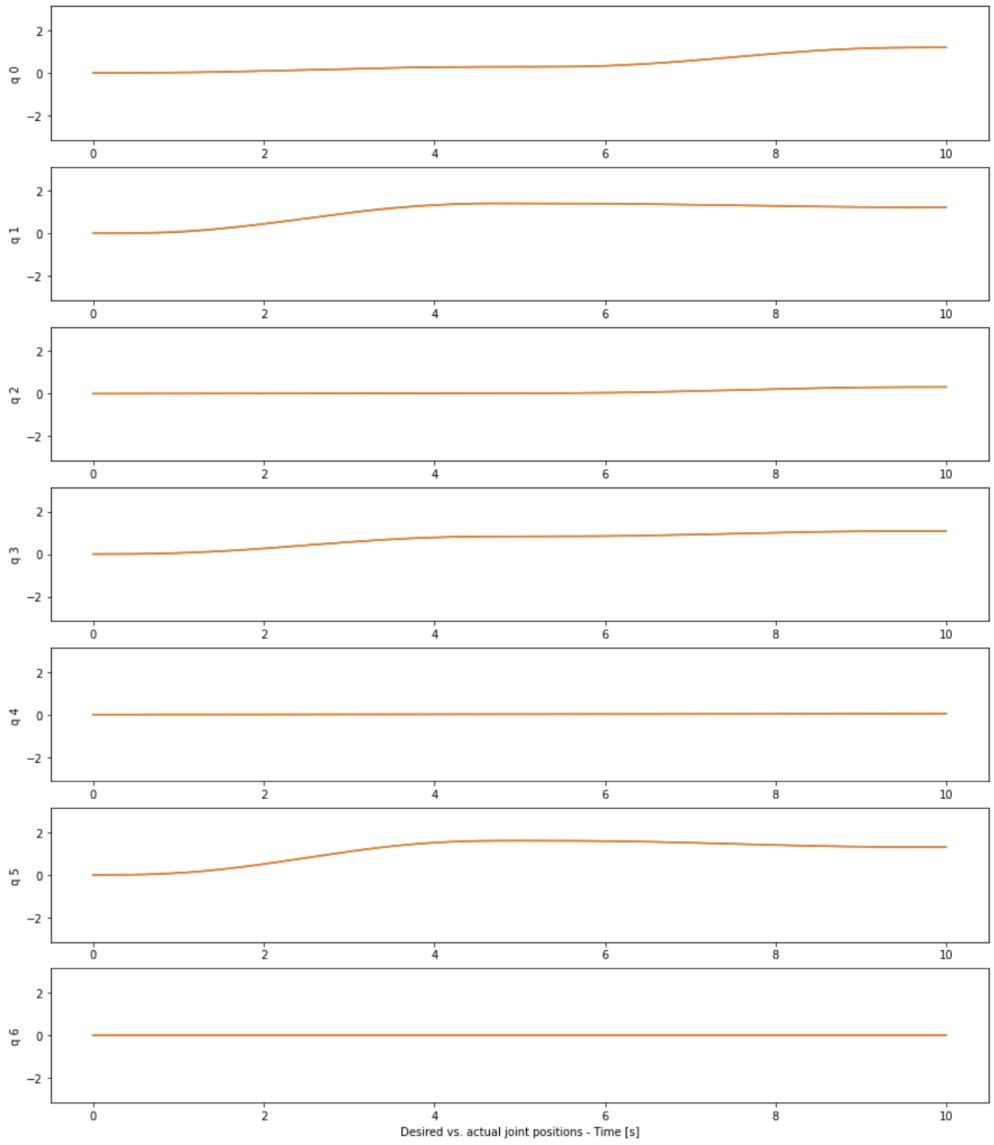

Figure 5: Desired and actual joint positions

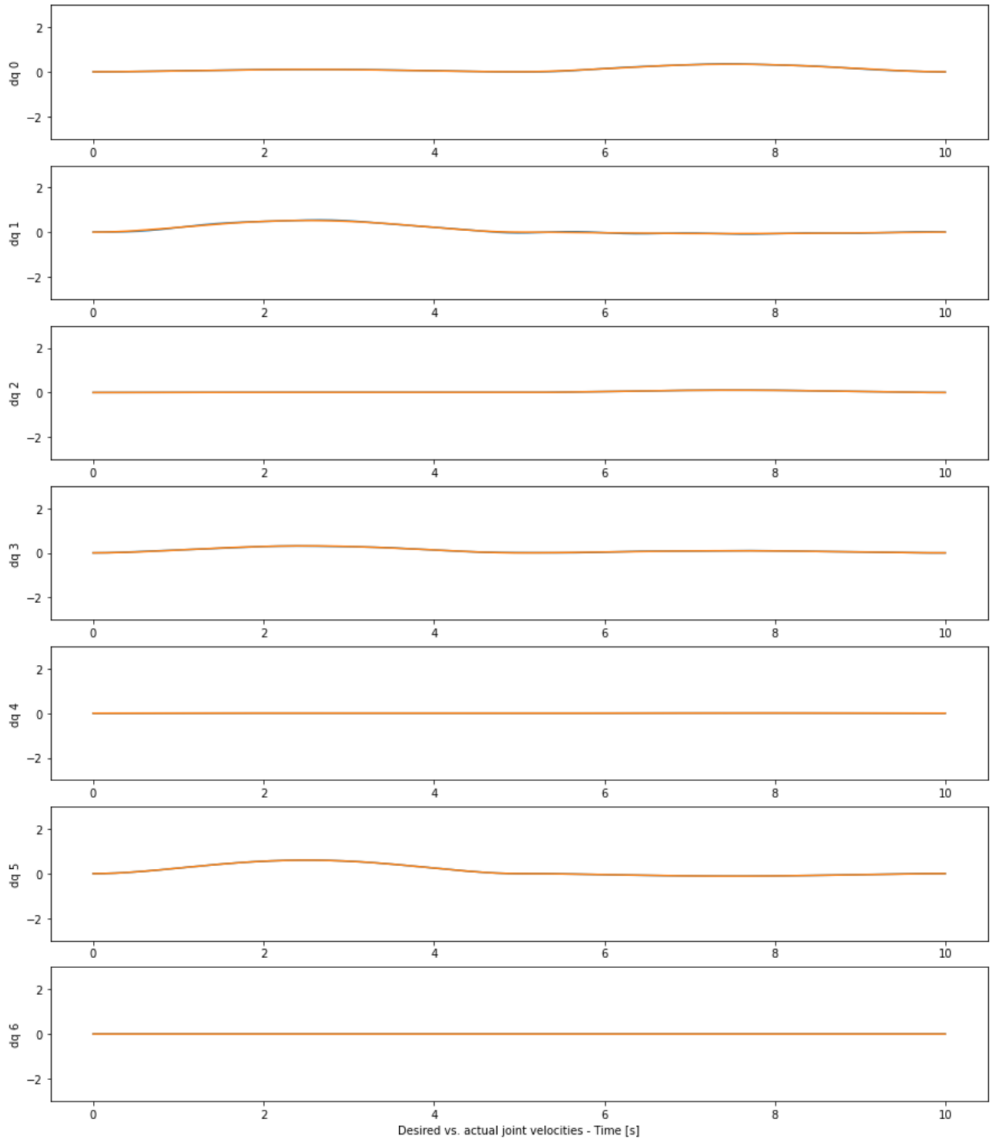

Figure 6: Desired and actual joint velocities

The desired and actual end-effector positions and velocities are plotted below:

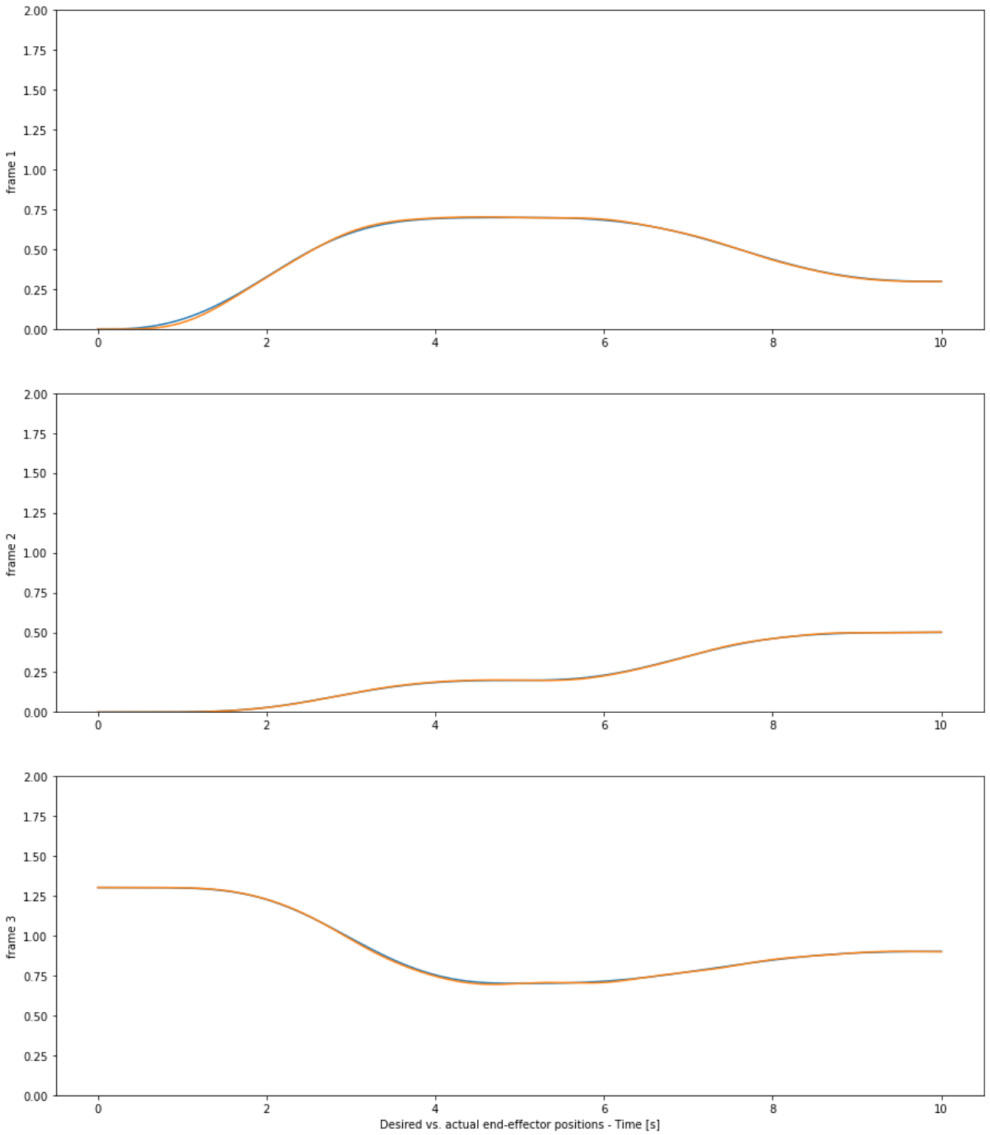

Figure 7: Desired and actual end-effector positions

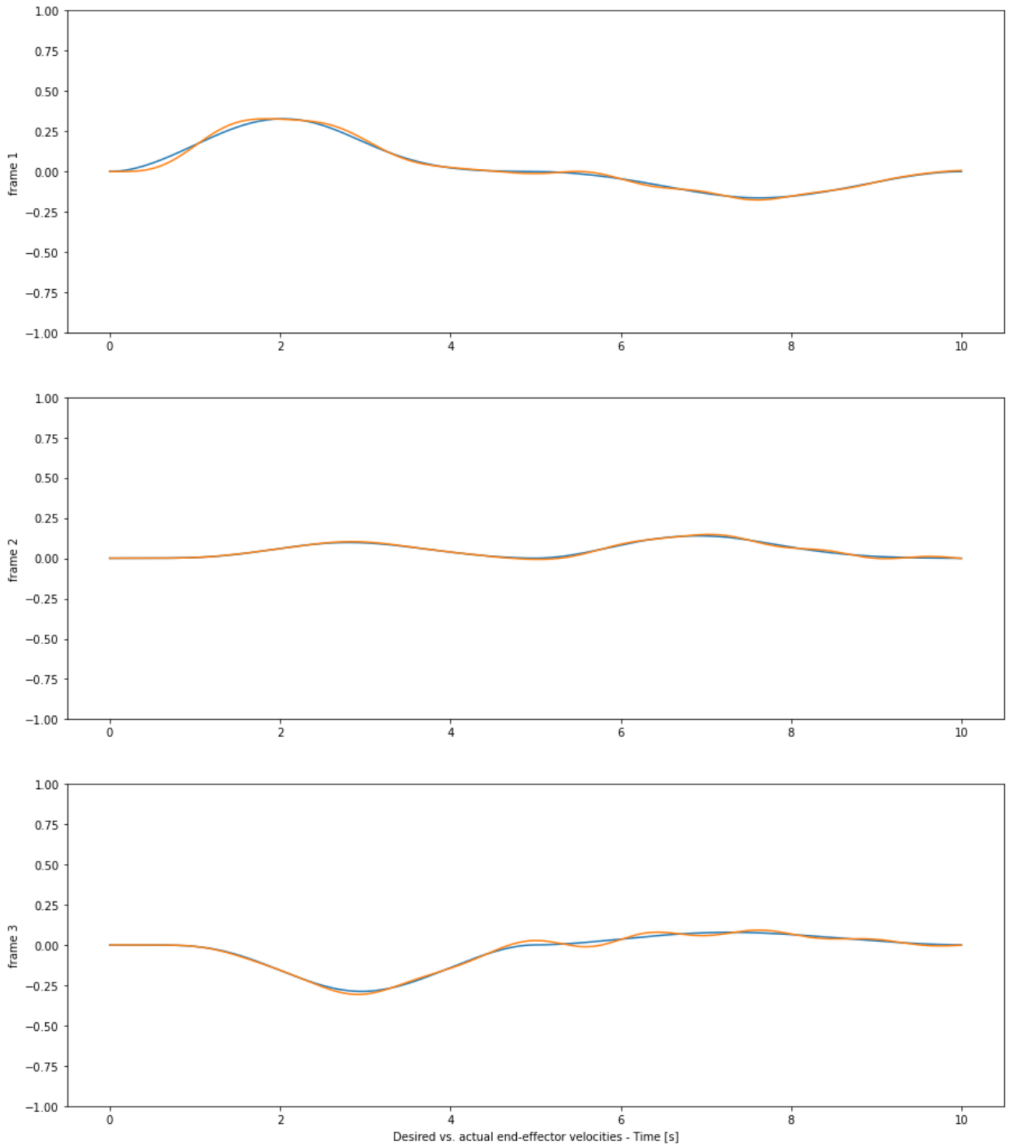

Figure 8: Desired and actual end-effector velocities

3. We can notice that the robot can follow the desired variables with a tolerable error. But we have no idea what the actual trajectory of the end-effector is because for question 2 we only control in the joint angle space.

Question 3

1. Resolved-rate control allows us to perform direct control of the robot's end-effector velocity. We do not need to use inverse kinematics algorithm every iteration step and instead recompute the inverse of Jacobian matrix at each step. Related methodology is use the inverse of Jacobian to convert the end-effector velocity to the joint velocity first by:

   $$\dot{\theta} = J^{\dagger}\dot{x}$$

   and then set the desired joint velocity as:

   $$\dot{\theta}_{desired} = J^{\dagger}(P(x_{desired} - x) + \dot{x}_{desired})$$

   The desired joint torque is:

   $$\tau = D(\dot{\theta}_{desired} - \dot{\theta})$$

   P and D are coefficient matrix. They are diagnoal matrix (P is 3×3 and D is 7×7) and needed to be tuned to get a satisfying control effect. Here we choose $P = diag(30)$ and $D = diag(4)$.

2. The desired and actual joint angle positions and velocities are plotted below. The yellow line represents the actual variable and the blue line represents the desired variable.

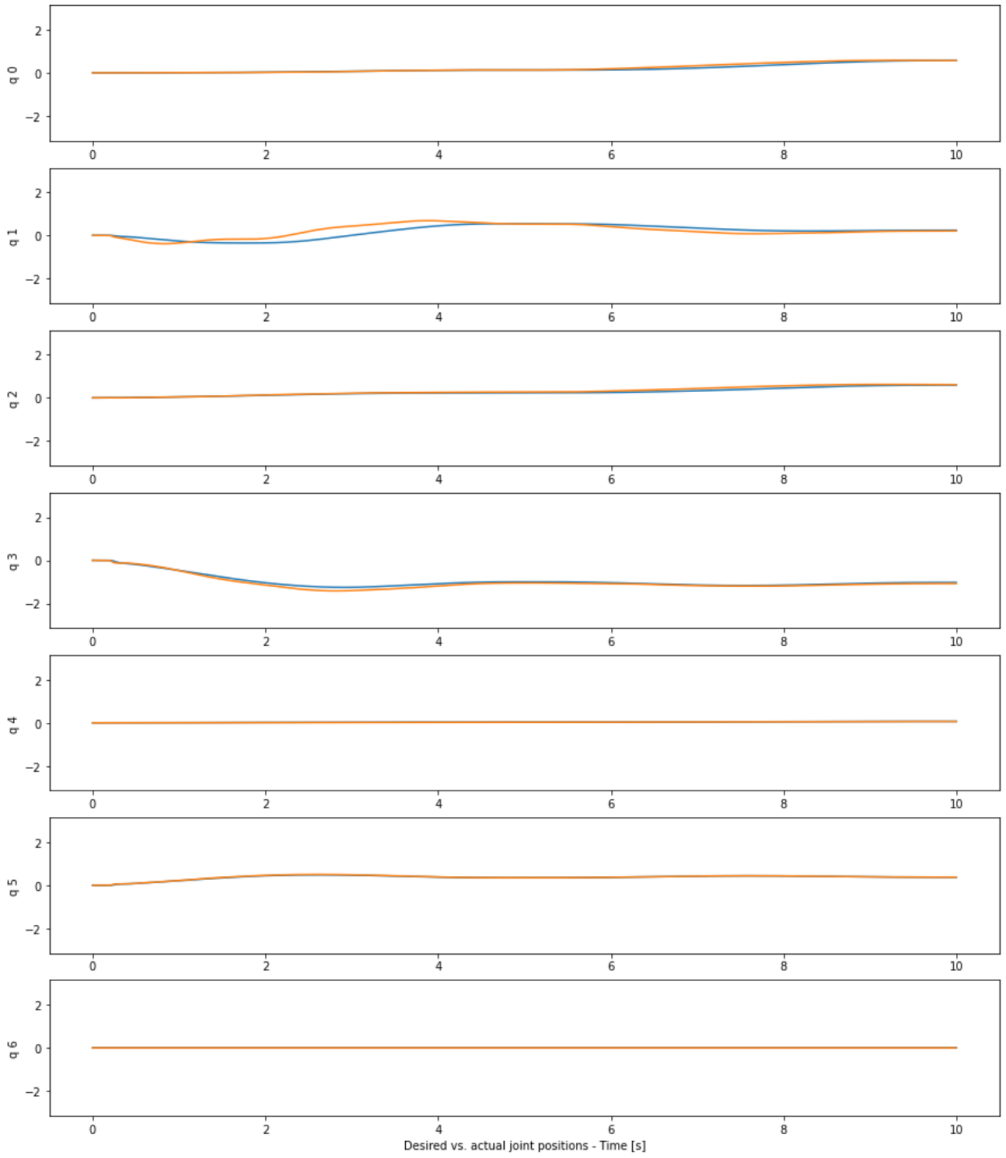

Figure 9: Desired and actual joint positions

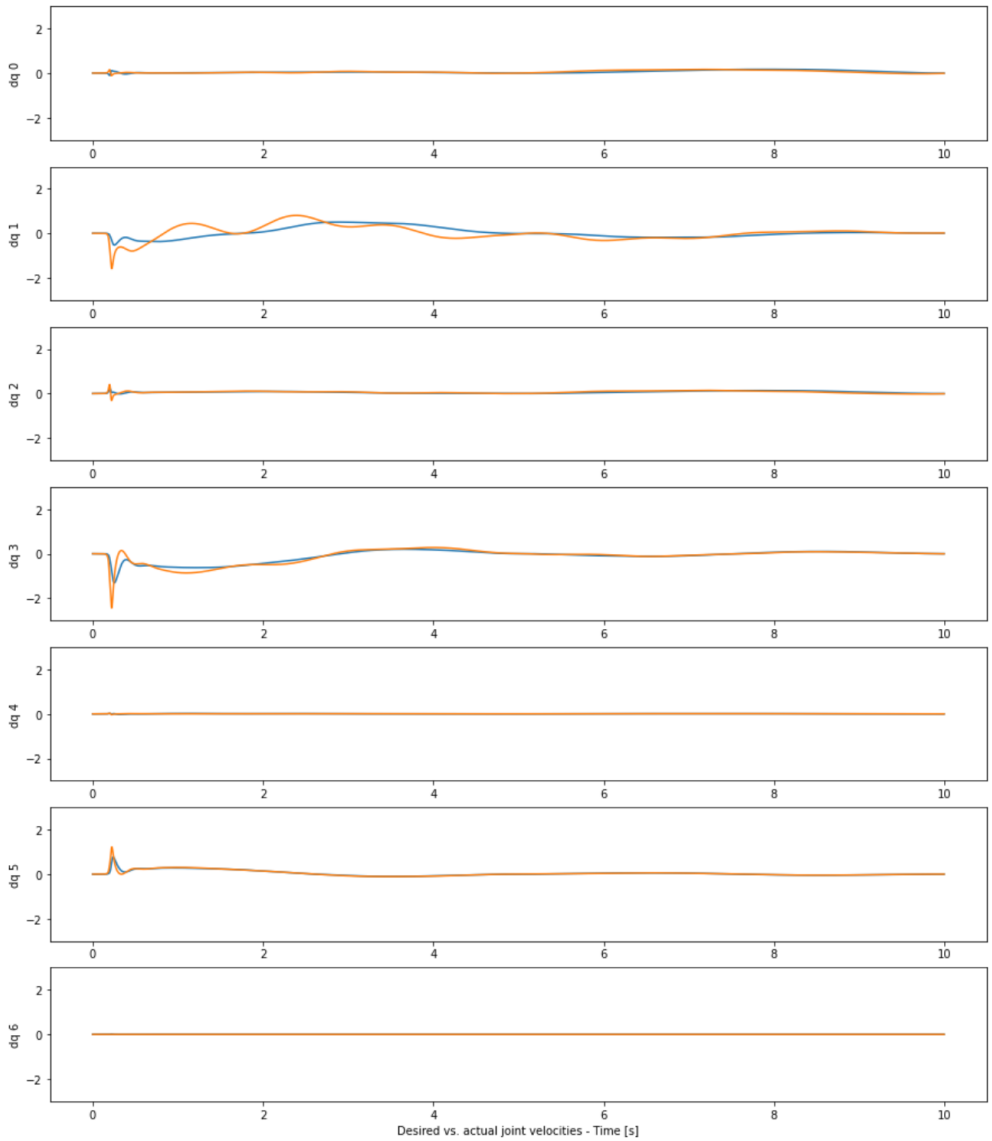

Figure 10: Desired and actual joint velocities

The desired and actual end-effector positions and velocities are plotted below:

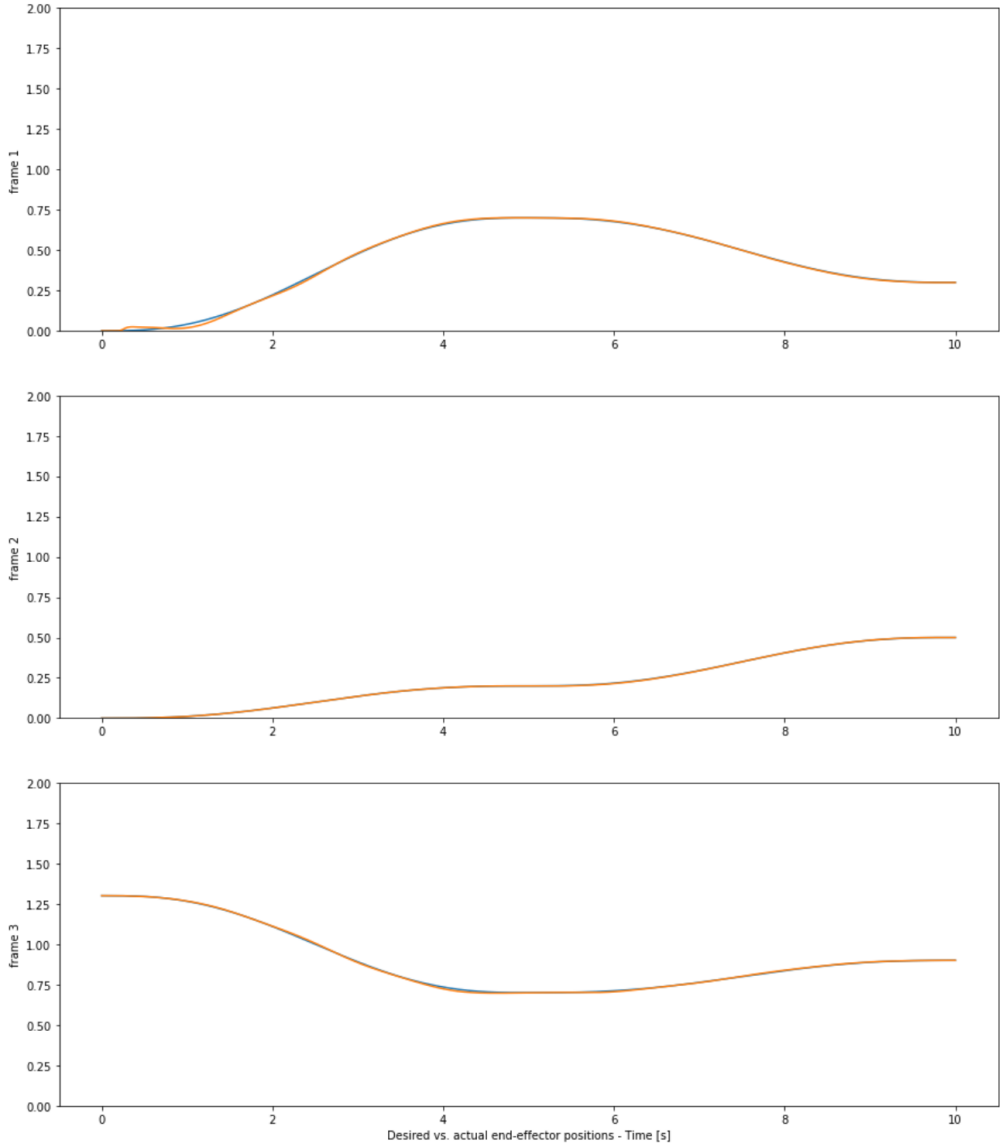

Figure 11: Desired and actual end-effector positions

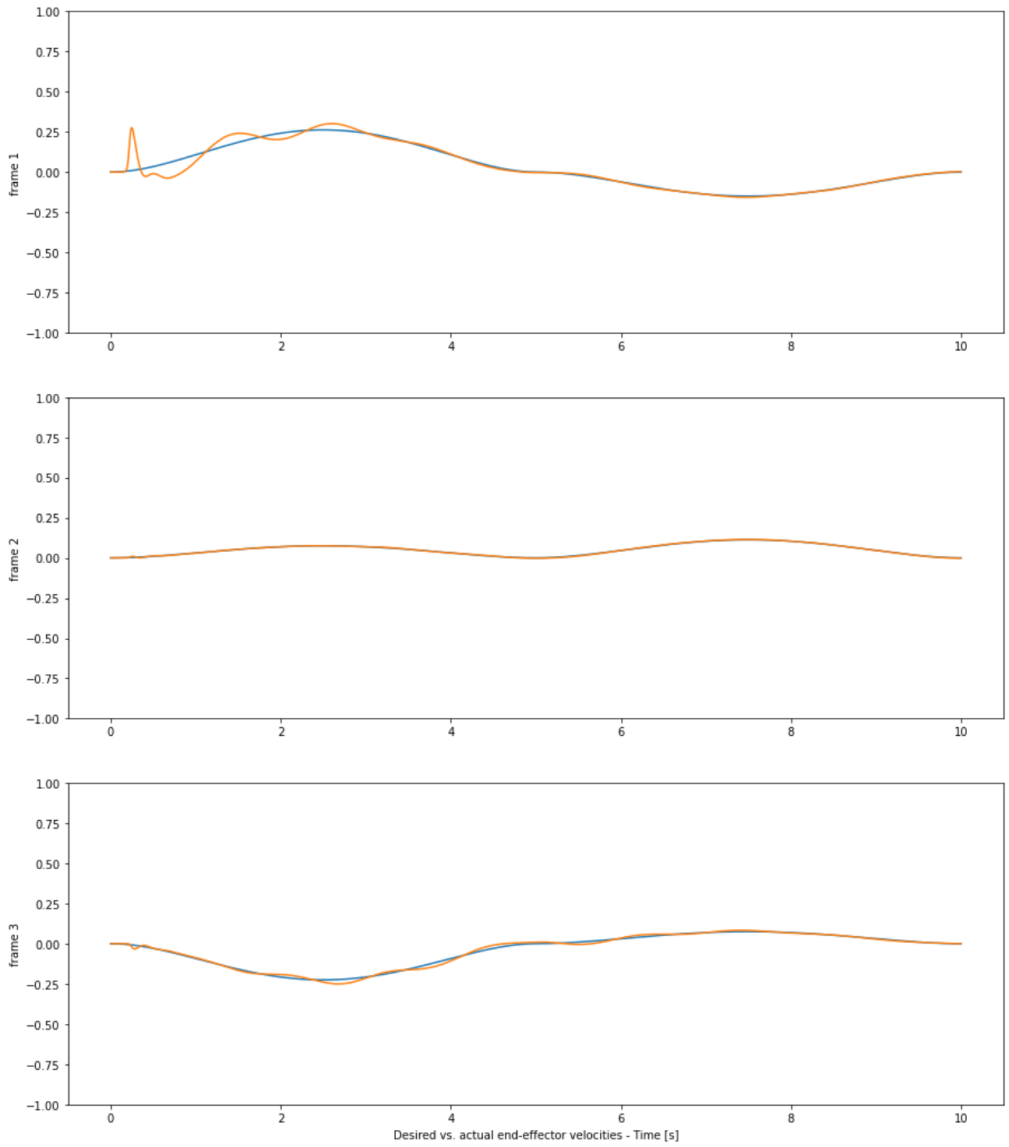

Figure 12: Desired and actual end-effector velocities

3. We can notice that compared to question 2, there are some error when the robot was tracking the desired variables. Since we perform the end-effector space control in question 3 , we can ensure that the end effector to follow a straight line trajectory when moving but have no idea about the joint variables.

4. After we adding a desired joint configuration and the nullspace projector, similarly to question 1, the robot will reach each goal every 5 seconds and meanwhile trying to get as close as possilble to the desired configuration.

Question 4

1. The methodology of impedance control with gravity compensation is:

$$\tau = J^T(P(p_{des} - p_{measured}) + D(\dot{p}_{des} - \dot{p}_{measured})) + G(\theta)$$

The gravity term $G(\theta)$ can be derived from the Recursive Newton Euler Algorithm (RNEA):

$$RNEA(\theta, \dot{\theta}, \ddot{\theta}) = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = \tau$$

We can get the gravity term by simply setting the joint velocity and acceleration to 0.

$$G(\theta) = RNEA(\theta, 0, 0)$$

The advantage of using impedance control is that we only need to compute the transpose of the Jacobian which is more computationally efficient and can also naturally avoid the singularity problem when computing the pseudo inverse of the Jacobian matrix.

2. The desired and actual joint angle positions and velocities are plotted below. The yellow line represents the actual variable and the blue line represents the desired variable.

Here we choose $P = diag(850)$ and $D = diag(10)$.

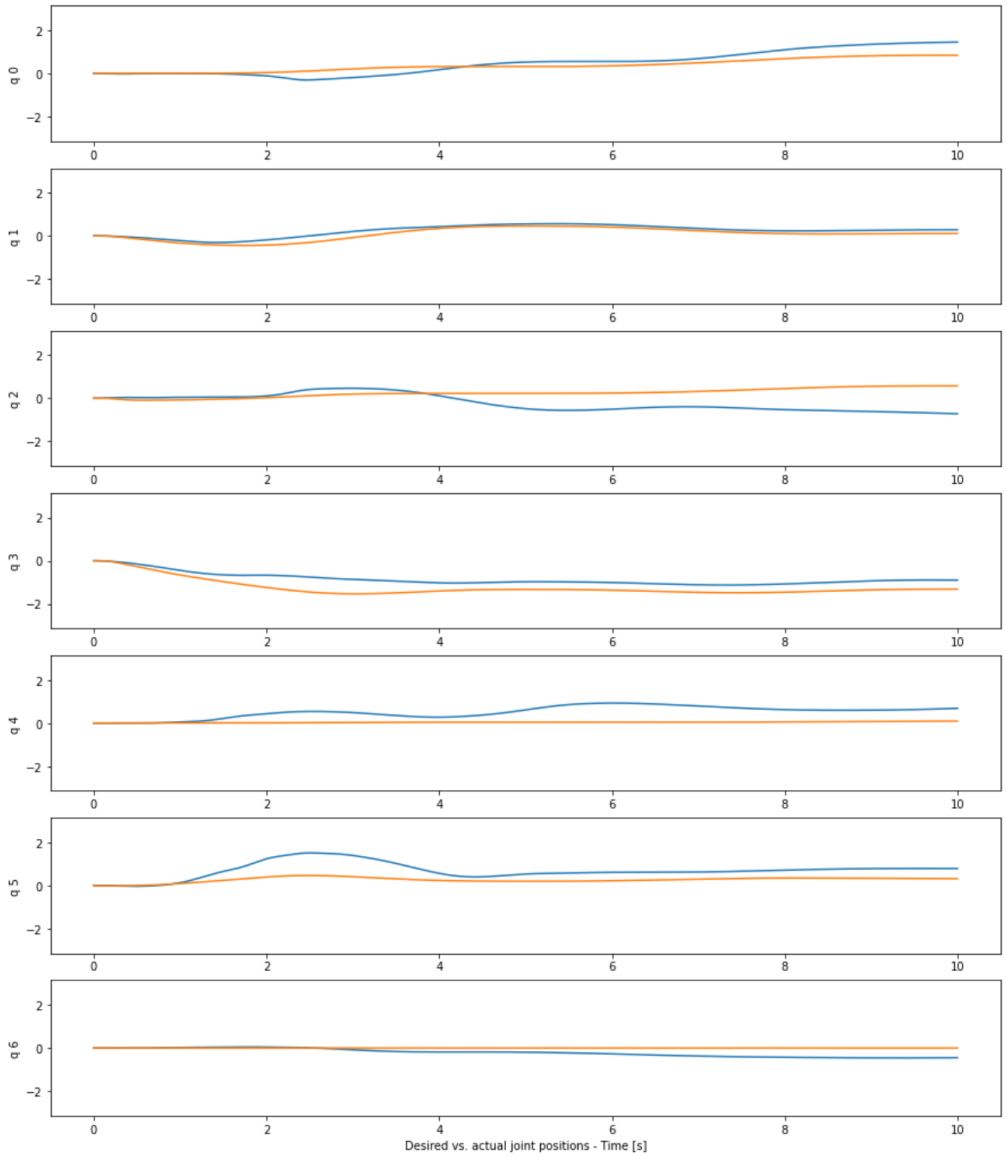

Figure 13: Desired and actual joint positions

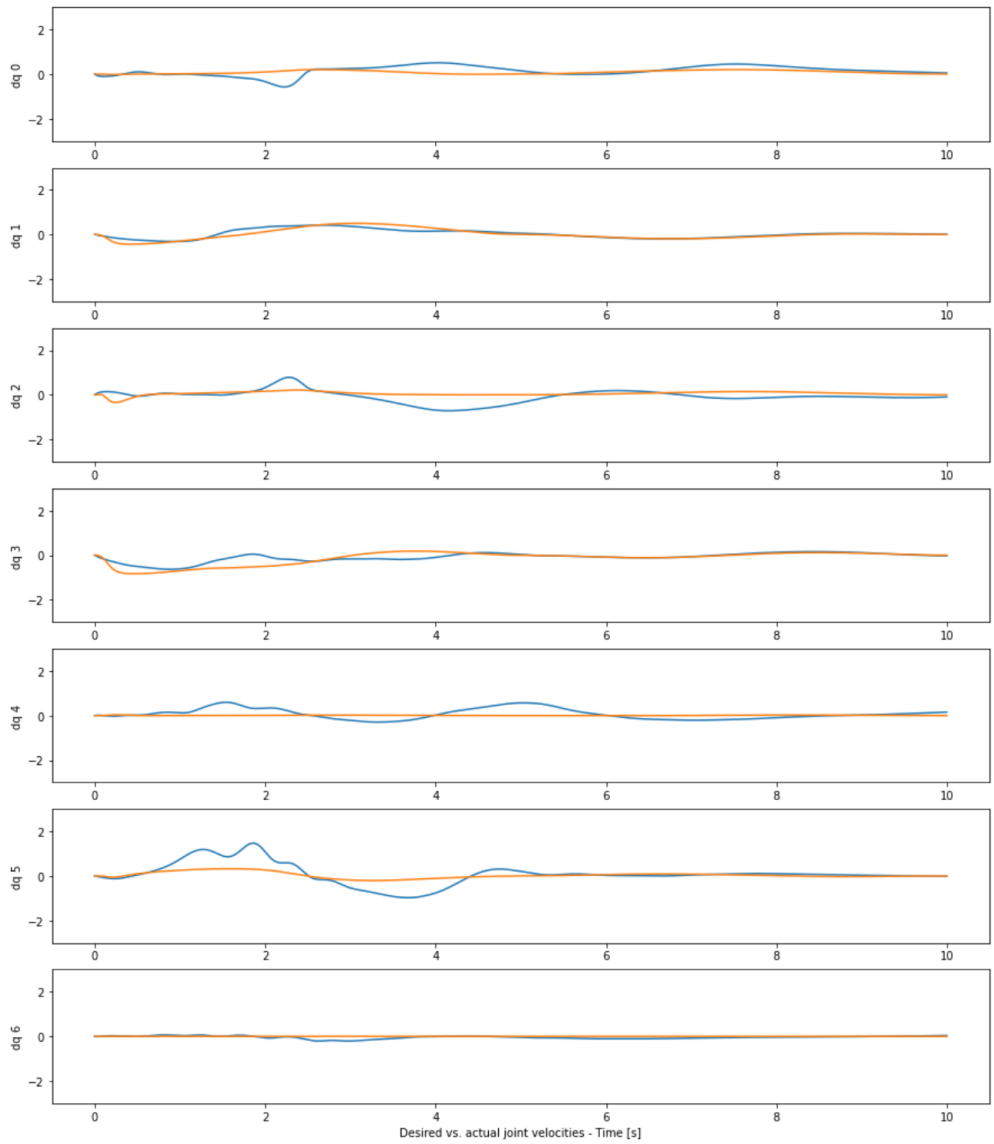

Figure 14: Desired and actual joint velocities

The desired and actual end-effector positions and velocities are plotted below:

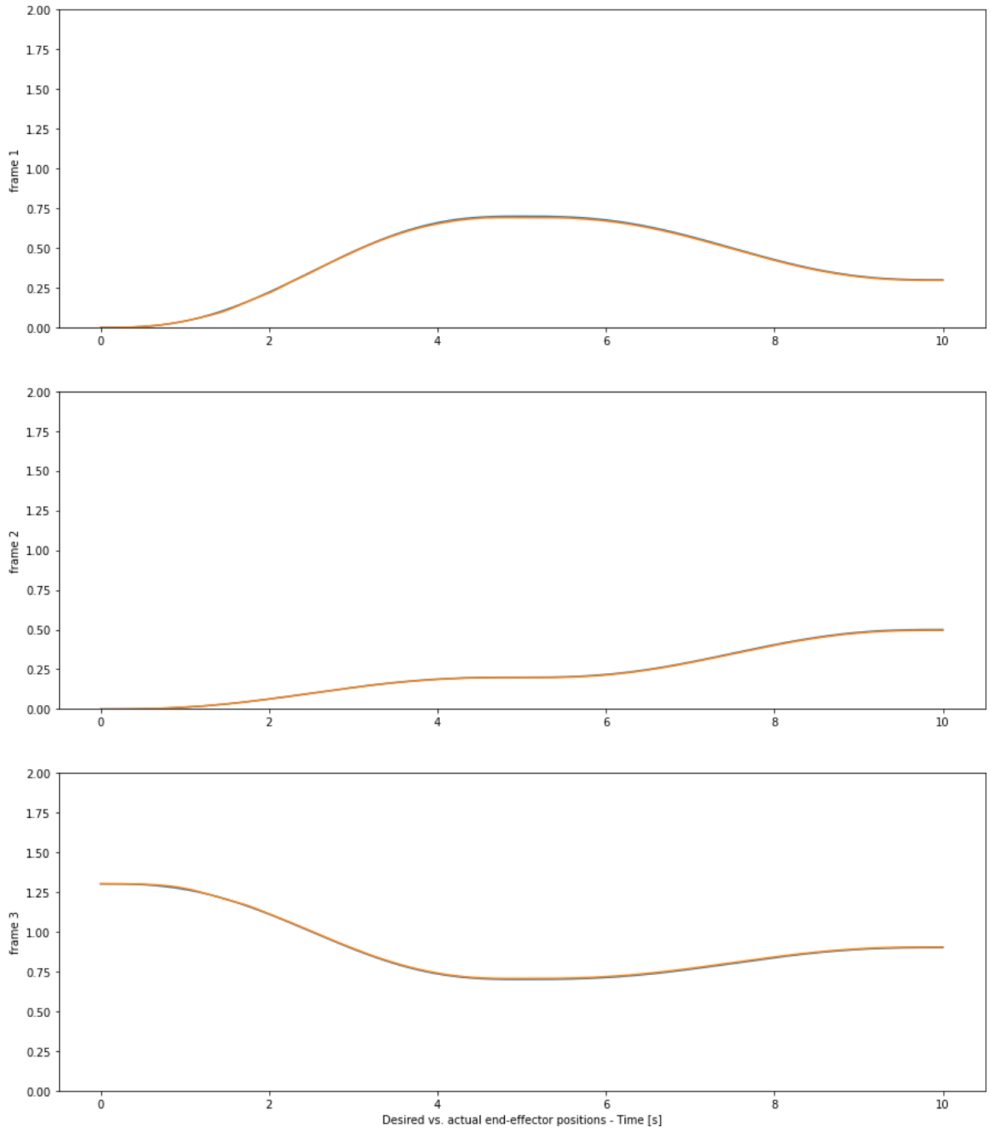

Figure 15: Desired and actual end-effector positions

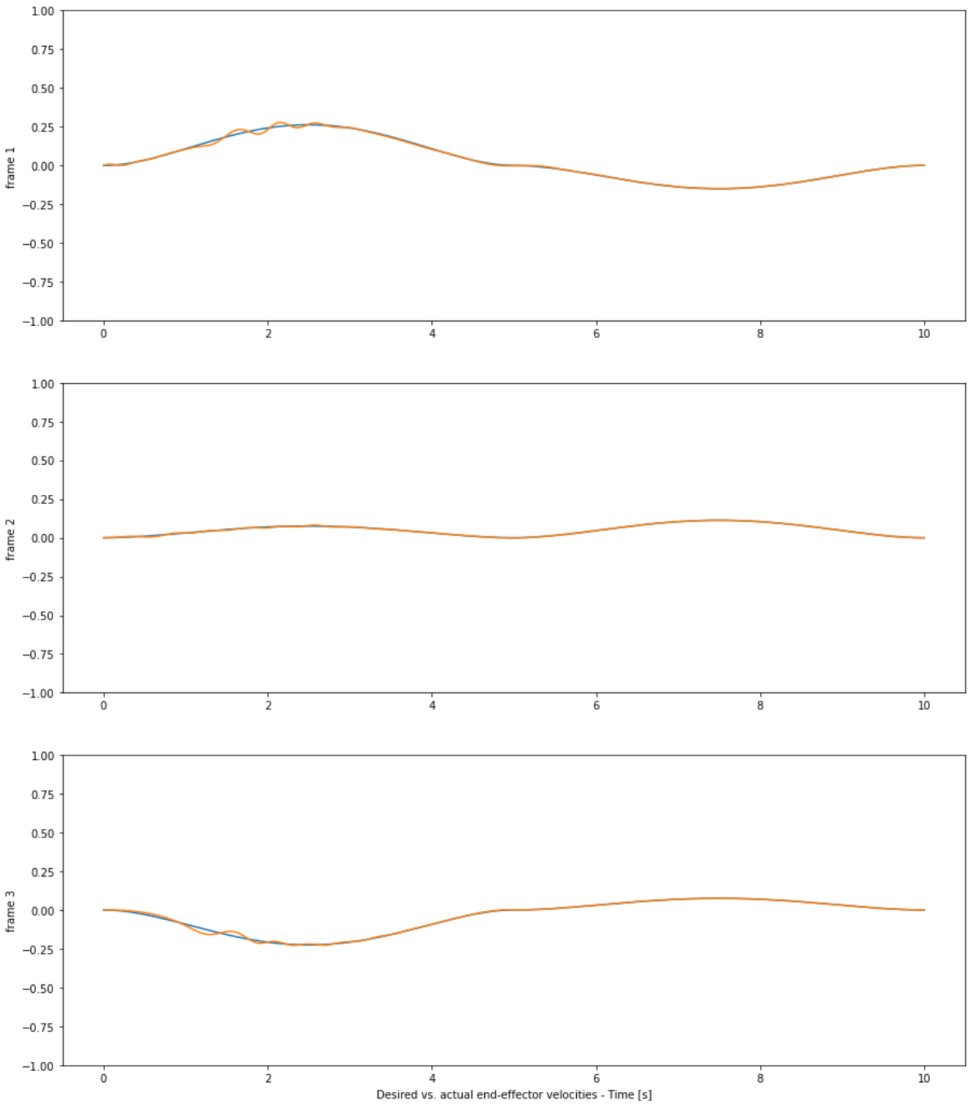

Figure 16: Desired and actual end-effector velocities

3. The joint damping factor here we choose is -0.1. The reason why we set it a negative number is to simulate the robot closer to a robot in physical world. When an actual robotic arm is moving, the friction and viscosity exist in every joint which causes the waste of energy. If we set this value to 0, the simulated robot model will do a lot of redundant joint motion although we can still tracking the goal in end-effector space.

   Please check the attached media file for a comparison of impedance controller with and without the damping factor. The corresponding plots are listed below:

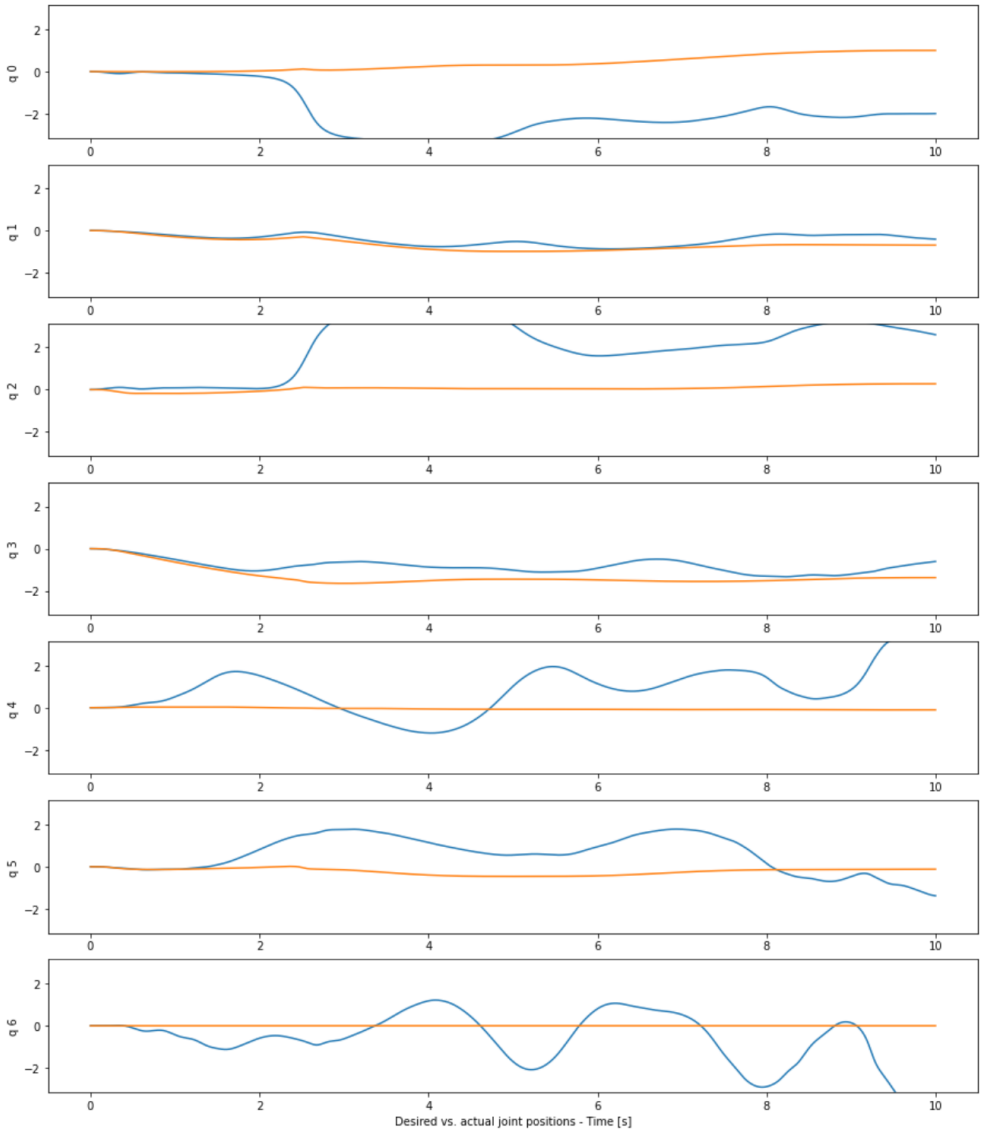

Figure 17: Desired and actual joint positions (without damping factor)

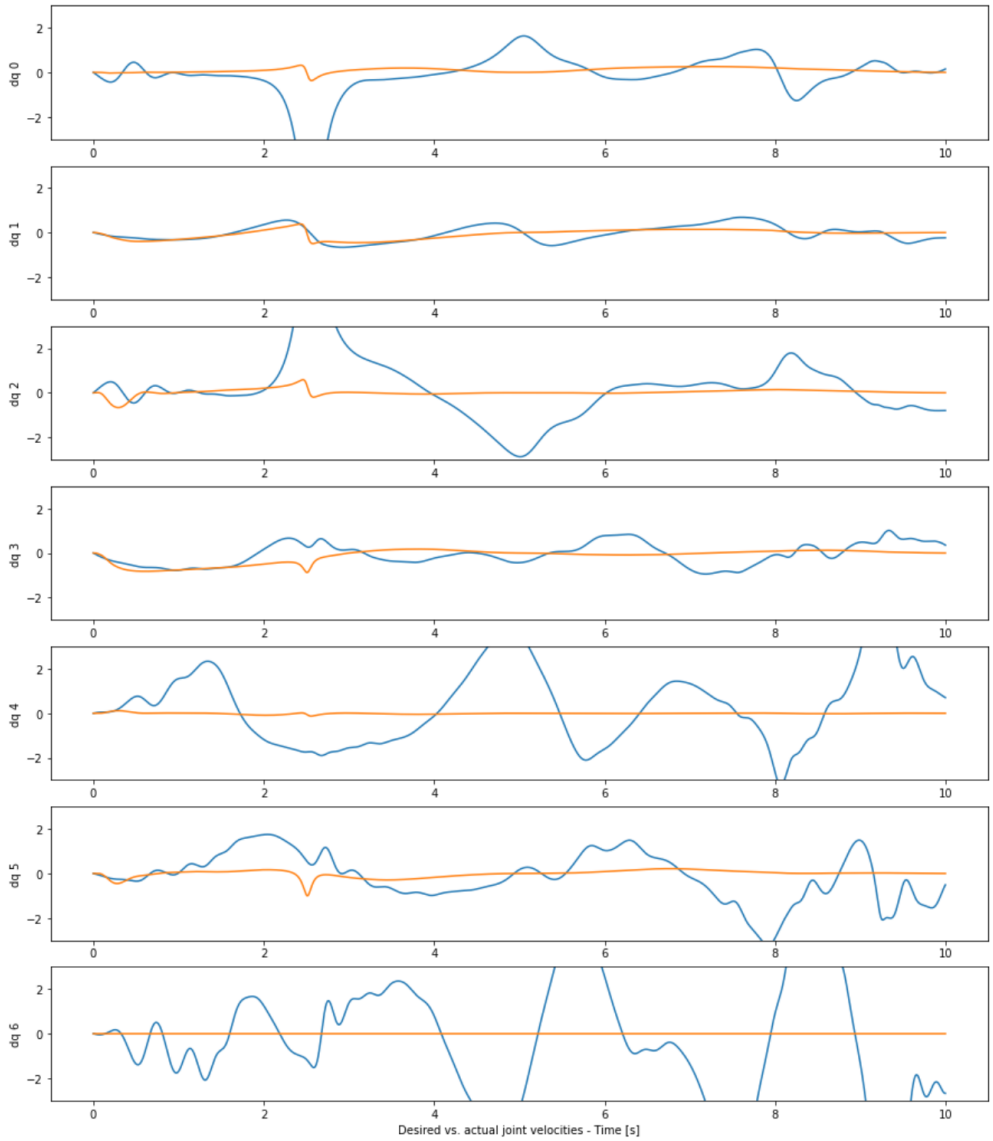

Figure 18: Desired and actual joint velocities (without damping factor)

The desired and actual end-effector positions and velocities are plotted below:

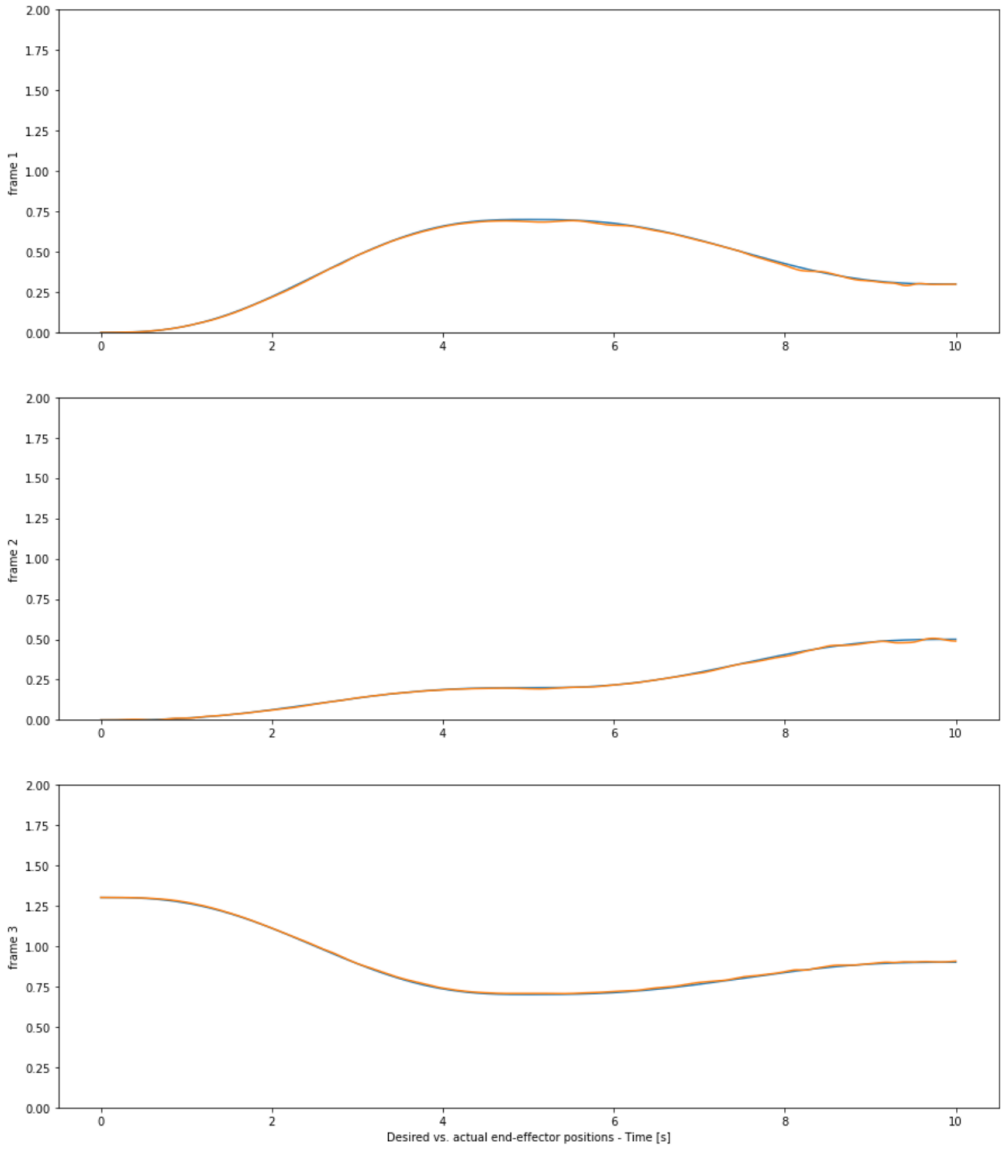

Figure 19: Desired and actual end-effector positions (without damping factor)

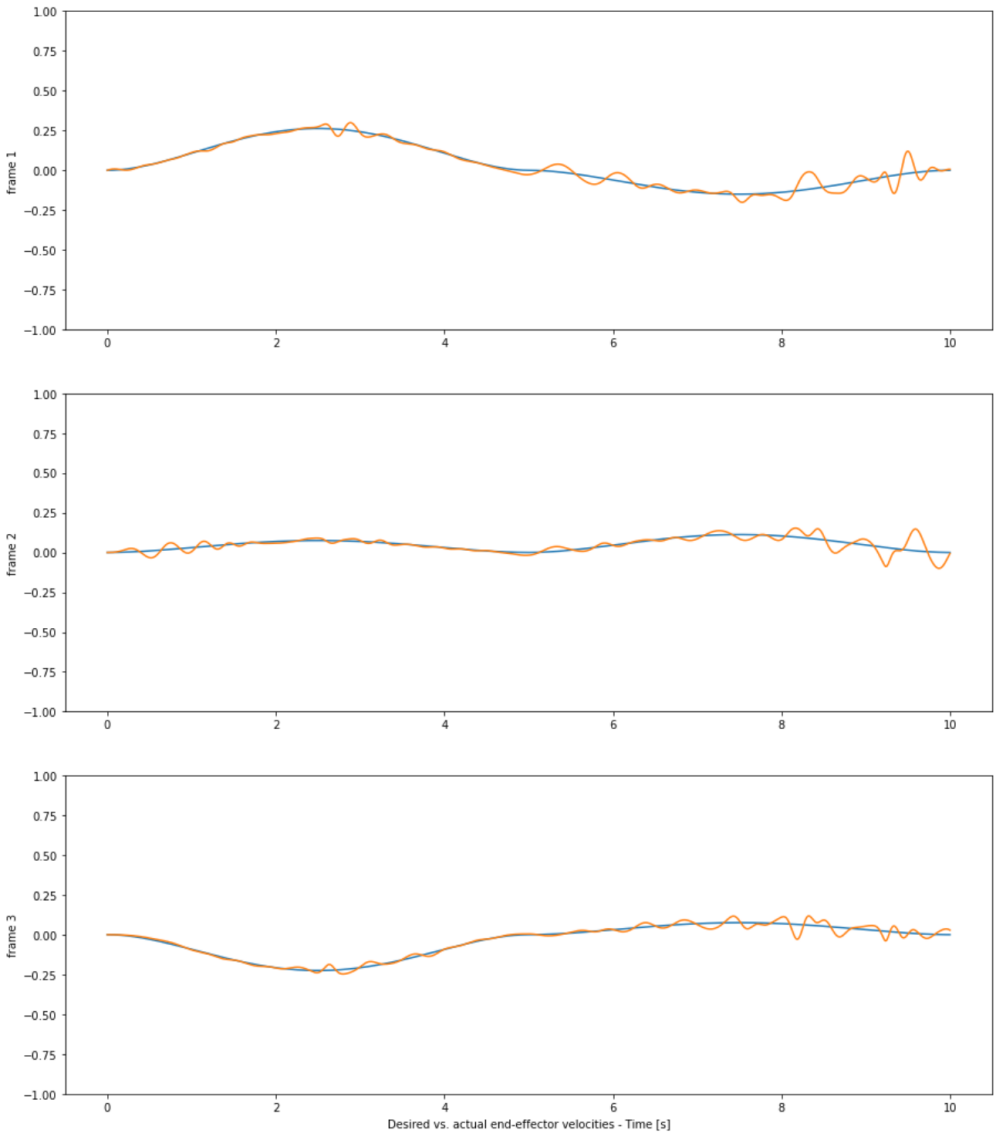

Figure 20: Desired and actual end-effector velocities (without damping factor)

4. Compared to question 2 and 3, I would more like to choose the impedance controller with gravity compensation and damping factor to complete this moving task. Here are some reasons:

   First, impedance controller can track the goal accurately as we desired with the polynomial interpolation. When we use the joint space controller, we actually have no idea of the trajectory in the end-effector space. Impedance controller with polynomial interpolation can fix this problem.

   Second, when using the impedance controller, we just have to compute the transpose of the Jacobian matrix compared to the resolved rate control which need to compute

the pseudo inverse of the Jacobian. The computation of transpose is quite faster than the pseudo inverse so that it is more time-efficient. A more important point is we do not have to consider the singularity problem which is a crucial and annoying problem when we compute the pseudo inverse.

The last reason is, the impedance controller is more comprehensive since it is designed to make the robot more compliant to the external forces. The joint space controller and resolved rate controller only consider the aspect of kinematics either in joint space or the end-effector space. The impedance of the robot is how it will react when interact with the surrounding environment. By controlling the impedance of the robot, it can act more precisely. For example, in position control, if we are far from the goal, the difference will generate a very large force on the robot which can be very dangerous to people near the robot. In impedance control, we can control the motion and force of the robot at same time thus it can avoid some damaging effect and achieve our goal smoothly.