Thanki Dhruv A
UD.ID: 702529633

Intro to Robotics Project:

DH Parameter:

Links	**d**	**theta**	**alpha**	**a**
b-0	0	pi/2	pi/2	0
0-1	340	q1	-pi/2	0
1-2	0	q2	pi/2	0
2-3	400	q3	pi/2	0
3-4	0	q4	-pi/2	0
4-5	400	q5	-pi/2	0
5-6	0	q6	pi/2	0
6-e	126	q7	0	0

End-Effector Pose:

To find the pose of the end-effector, we do forward dynamics considering every link in the dh parameter given above. The link b-0 was used because the robot was in 0th frame while the obstacle coordinates were given in base frame, hence the translation is needed to be pre-multiplied to bring the robot to the base frame.

$$T_e^b = T_0^b * T_1^0 * T_2^1 * T_3^2 * T_4^3 * T_5^4 * T_6^5 * T_e^6$$

The pose vector is created using the position of the end-effector and the euler angles of the end-effector for orientation calculated from the rotation matrix of the end-effector.

The pose of the given initial condition is taken as the first point of the trajectory(i.e. qO) and the pose of desired position and orientation is calculated to compute the error term. All the poses were calculated using the forward.m function. Analytical Jacobian was also calculated using end-effector pose from this function.

Thanki Dhruv A
UD.ID: 702529633

Project Breakdown:

Part 1: Jacobian Pseudo-Inverse

First step was to generate code so that the end-effector reaches the desired position with desired orientation with the help of pseudo-inverse jacobian of the error term which is the difference between current pose and desired pose.

$$\Delta q = K * J_A^+ * (x_d - x_i)$$

Part 2: Visualising in a physics simulator

Once the manipulator is able to converge to the desired pose, a communication link was set up such that the iterative configurations generated while converging were streamed to the simulator for visual validation. Here we used CoppeliaSim (V-rep) Simulator. Once visual confirmation is achieved from the simulator, we set up the obstacle in the simulator according to the size and position given in the problem statement. After the simulator is set up we check where the collisions take place.

At this point the manipulator is able to move from initial pose to final pose without any specified trajectory but it is colliding with the obstacle.

Part 3: Adding Waypoints

First we try to find an intuitive trajectory which would cause fewer problems though this is usually hard to do. But we could add carefully selected points in the workspace as waypoints for the manipulator to pass through.

1st Choice of Waypoints:
w1 = [1.100, 0.3576, 0.4660];
w1 = [1.100, -0.4189, 0.4660];

This leads to boundary singularities.

2nd Choice of Waypoints:
w1 = [0.9099, 0.3576, 0];
w2 = [0.9099, -0.4189, 0];

This choice did not lead to any singularity or collision but when the arm reached any of its waypoints, It took a lot of time to begin to converge to the next waypoint.

Thanki Dhruv A
UD.ID: 702529633

Part 4: Trajectory planning

Because of the long time that it took to converge to any of the waypoints, I decided to write a trajectory that consists of 3 equations of straight lines that passed through these waypoints all in one horizontal plane, 0.9099m high. The total length of the path was divided into 3000 points and the configuration of joint space at each of these points were calculated using jacobian pseudoinverse. All these configurations were fed to the simulation for a visual confirmation. One of the three equations is shown below:

```
function q = trajectory2(t)

q = [];
if t<=1000
    q = [0.9099,0.3576,0.4660];
    q = q - [0,0,(4.66*10^(-4))*t];
end
if t>1000 && t<=2000
    q = [0.9099,0.3576,0];
    q = q - [0,(7.765*10^(-4))*(t-1000),0];
end
if t>2000 && t<=3000
    q = [0.9099,-0.4189,0];
    q = q + [0,0,(4.66*10^(-4))*(t-2000)];
end
q = [q,-0.0000,1.5758,1.5661];
end
```

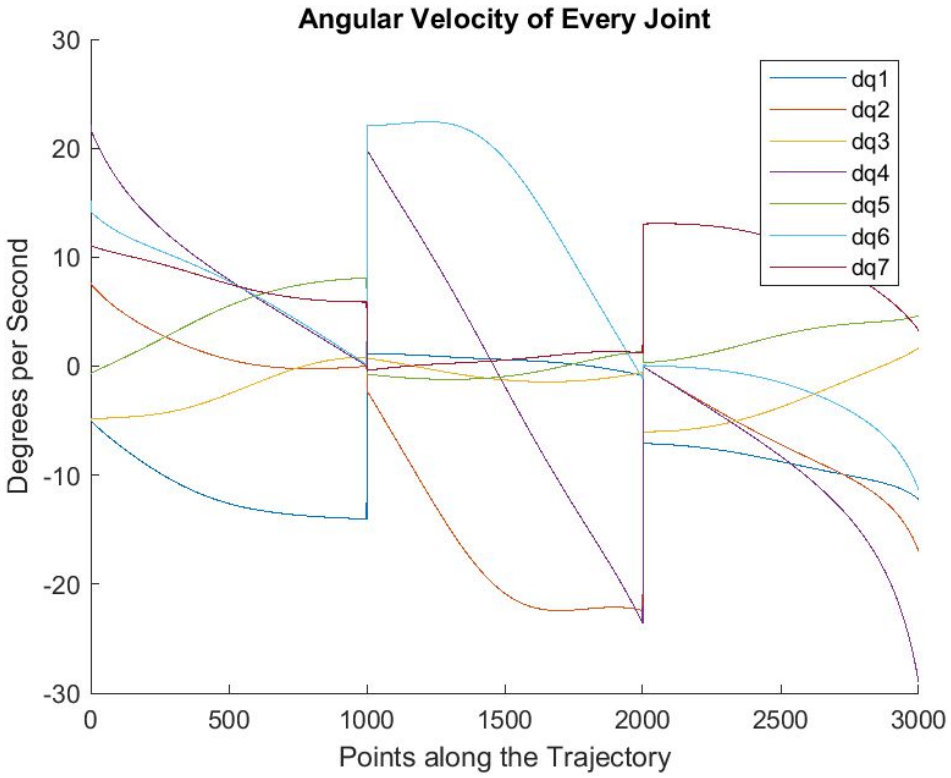
In this trajectory there were no collisions that took place and the joint limits were not exceeded and the joint angular velocities were also not exceeded. The error in orientation of the end-effector was also plotted to see the maximum error. All the graphs are shown below for validation.

Part 5: Obstacle and Singularity Avoidance

If the waypoints method were to fail and still the collision takes place, we can add terms to the numerical integration term. This additional increment in joint rotations is calculated such that it has no effect on the motion of the end-effector but moves the intermediate joints in such a way that the obstacle is avoided(Jacobian of this term is mapped to the null space). This addition of extra rotations could cause the joint to rotate beyond their limits and hence may require us to add another term to numerical integration term to avoid exceeding the joint limits and also angular velocities.

Thanki Dhruv A
UD.ID: 702529633

Validation:



Thanki Dhruv A
UD.ID: 702529633