CSCI 545 HW3

Rahul Krupani - 6585165228

October 22, 2023

1. **First, implement the forward kinematics function in the file fk.py. The function should receive the joint values q and the lengths of the 3 links L, and it should return the position of the robot's end-effector. Report the results for the following values:**

   (a) $\mathbf{q} = [0.0, 0.0, 0.0], \mathbf{L} = [1.0, 1.0, 1.0]$

   (b) $\mathbf{q} = [0.3, 0.4, 0.8], \mathbf{L} = [0.8, 0.5, 1.0]$

   (c) $\mathbf{q} = [1.0, 0.0, 0.0], \mathbf{L} = [3.0, 1.0, 1.0]$

   Since we only need the position, not the whole pose, we can simply use the formula

$$x = \Sigma_i^n \ell_i cos(\Sigma_j^i q_j)$$

$$y = \Sigma_i^n \ell_i sin(\Sigma_j^i q_j)$$

   This gives us the solutions for the questions (assuming q is in radians):

(a) (3.0, 0.0, 0.0)

(b) (1.217, 1.556, 0.0)

(c) (2.702, 4.207, 0.0)

2. For the inverse kinematics computation, you will use the scipy.optimize package.[1] We will numerically compute a solution through the following function call:

```
solution = minimize(objective, q0, method='SLSQP', bounds=bnds)
```

The final solution is:
x1 = 0.8639618601362655
x2 = 0.006620449644537282
x3 = 2.2382197588037123

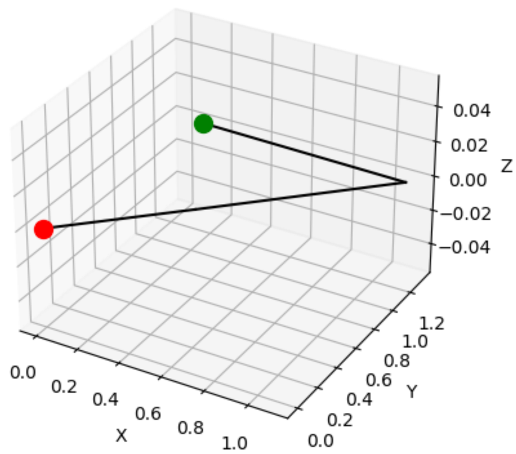


Figure 1: As you can see, the arm reaches the goal position

4. To ensure that the robot does not collide with the obstacle, we will add three obstacle collision constraints, one for each robot link. The constraints are provided as input to the optimization algorithm, as shown in the provided file **ik-b.py**. Fill in the code for the constraints, using the collision detection algorithm from the previous exercise. The parameters for the collision sphere are as follows:

$$\mathbf{c} = (0.6, 0.5, 0), r = 0.2$$

Note that if the constraints are satisfied, they should return a *non-negative* value. Using the following parameters for the obstacle, solve and visualize the solution. Save your visualization as `ik-b_solution.png`.

The final solution is:
x1 = 0.2512910125168038
x2 = 0.9149907547515956
x3 = 1.7363845790649686

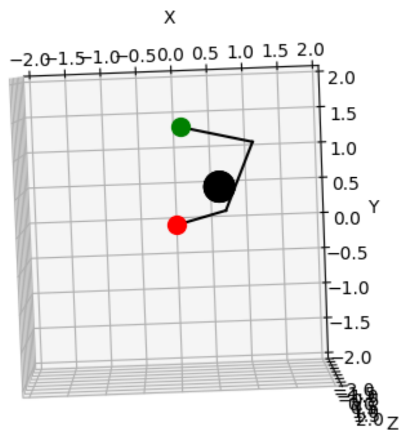
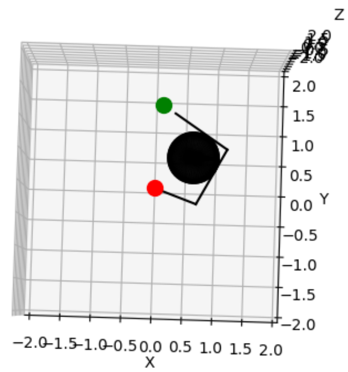


Figure 2: As you can see, the arm reaches the goal position while avoiding the obstacle

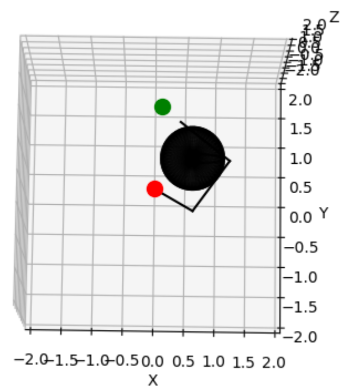
5. **Try increasing the radius of the obstacle r. What do you observe? Also experiment with different starting configurations $\mathbf{q_0}$. Discuss the results in a file named `answers.pdf`.**

While increasing the radius of the obstacle, you notice that this starting position only works until a radius of 0.4. After this point, the arm always collides since there is no room to go, this forms a concave region in the space (This is not a long-horizon planner).

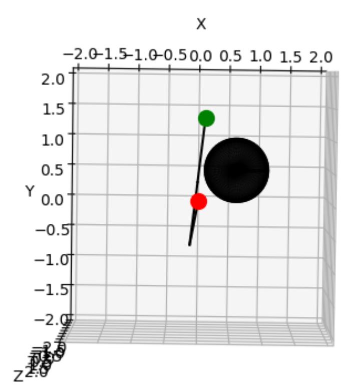
However, if you change the initial position, you can see that with certain positions, the arm can find a new path to the goal even with a larger radius. This shows the importance of the initial start point for any task.



(a) With radius 0.4m, the arm can still reach the goal



(b) With radius 0.5m, the arm collides and cannot reach goal



(c) With initial state $[\pi, 0, 0]$, the arm is able to find a new way to reach the goal