HW 3: Inverse Kinematics

TODO – Your Name Here

Instructions

- Complete all the questions, *including* the "Resources Consulted" question. We expect that most students will use about a paragraph, or a few bullet points, to answer that question, but you can go longer or shorter.
- Submit the PDF *and* the code separately on Gradescope (look for "HW3: PDF" and "HW3: Code").
- For the PDF: To make things simpler for us to grade / inspect, please answer the questions in order (resources consulted first, then question one, then two, etc.). There is no page limit, but (as a high-level piece of advice) avoid writing excessively long-winded solutions. Use LaTeX for this; you can build upon this file. If you do that, you can remove the text that describes the actual questions if you want, please just make it *really easy* for us to see which question you are answering.
- For the code: some questions have code, which you will need to write. **Please use Python 3** for your code. Just submit everything as a single `.zip` file. Please include a brief README or brief comments in the code that make it *very easy* for us to run.

Resources Consulted

Question: Please describe which resources you used while working on the assignment. You do not need to cite anything directly part of the class (e.g., a lecture, the CSCI 545 course staff, or the readings from a particular lecture). Some examples of things that could be applicable to cite here are: (1) did you collaborate with a classmate; (2) did you use resources like Wikipedia, StackExchange, or Google Bard in any capacity; (3) did you use someone's code? When you write your answers, explain not only the resources you used but HOW you used them. If you believe you did not use anything worth citing, *you must still state that below in your answer* to get full credit.

Answer:
coding:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html
https://stackoverflow.com/questions/3810865/valueerror-unknown-projection-3d

In this assignment, you will compute numerically the inverse kinematics of a planar robot with $n = 3$ rotational joints. Specifically, our goal is to compute the joint-positions that bring the end-effector to a desired position $\mathbf{p_d}$. In this exercise we will ignore the orientation of the end-effector. We will formulate the problem as an optimization program as follows:

$$\begin{aligned} \underset{\mathbf{q}}{\text{minimize}} \quad & \|FK(\mathbf{q}) - \mathbf{p_d}\|^2 \\ \text{subject to} \quad & l_i \leq q_i \leq u_i,\ i = \{1, \ldots, n\}. \end{aligned}$$

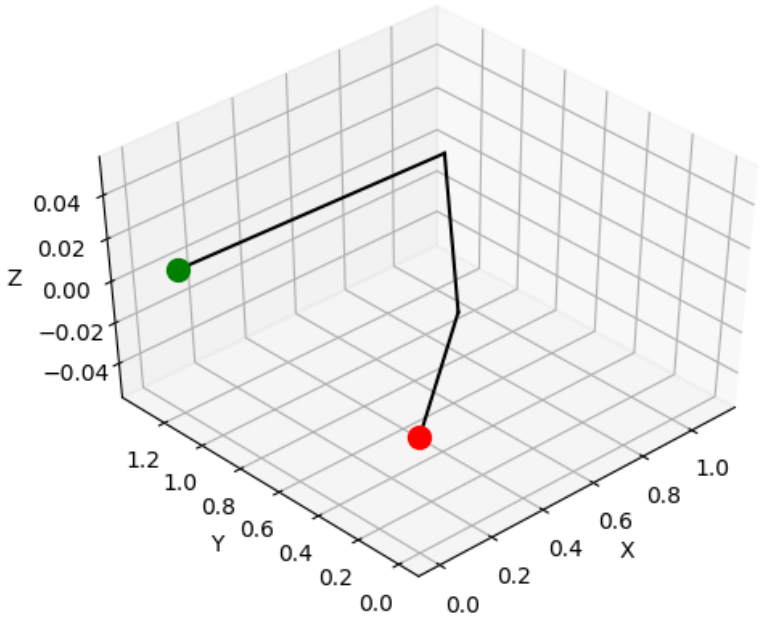
The objective is to find the values of the joints $\mathbf{q}$ that minimize the difference between the actual position of the end-effector computed using the forward kinematics $FK(\mathbf{q})$, and the desired position $\mathbf{p_d}$, subject to the joint limits l_i, u_i.

1. Report the results for the following values. Note that q is in radians.

```
A:
[3.  0.  0.]
B:
[1.21742749 1.55602    0.        ]
C:
[2.70151153 4.20735492 0.        ]
```

2. Visualize the result with the given plotting function. Save your figure as ik-a solution.png and add it to the PDF.

```
Initial SSE Objective: 1.8667613974260215
Final SSE Objective: 5.973916737861113e-10
Solution
x1 = 0.743128279036369
x2 = 0.20452802125737166
x3 = 2.1496030495325957
```

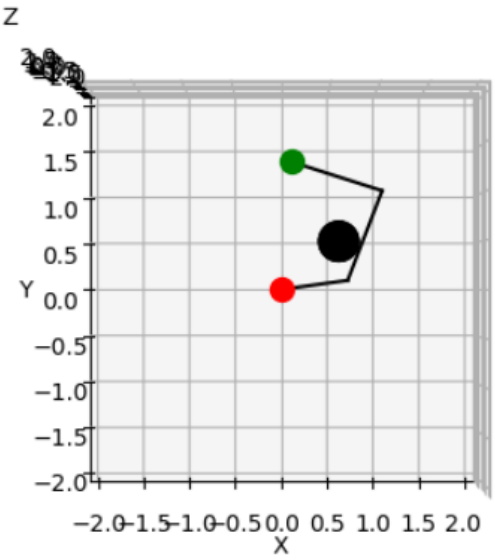


3. You will now add an obstacle that the robot should avoid. You will approximate the obstacle with a sphere, with circle $\mathbf{c} = (c_x, c_y, c_z)$ and radius r. To detect whether the robot collides with the obstacle, the easiest way is to implement a line-sphere intersection algorithm[1]. If there is no real solution, then there is no intersection between a line and the sphere. In the provided file **collision.py**, fill in the function `line_sphere_intersection`. It should implement the algorithm and return the value under the square-root (the discriminant).

[1] `https://en.wikipedia.org/wiki/Line-sphere_intersection`

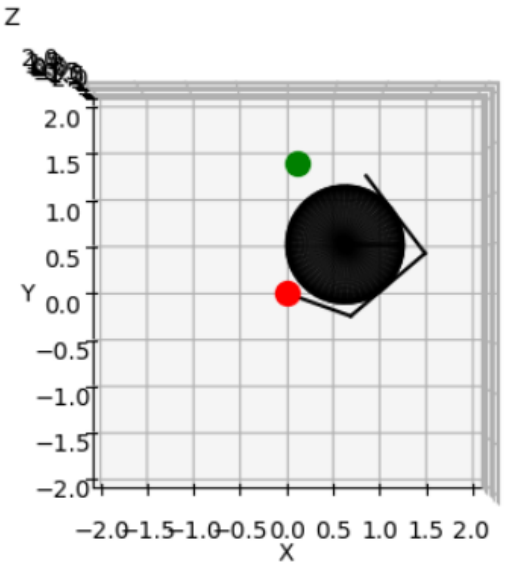
4. Using the following parameters for the obstacle, solve and visualize the solution. Save your visualization as ik-b solution.png and add it to the PDF.

```
Initial SSE Objective: 1.8667613974260215
Final SSE Objective: 4.546474863205245e-10
Solution
x1 = 0.13467820325123353
x2 = 1.0685020459637808
x3 = 1.6307402504821535
```



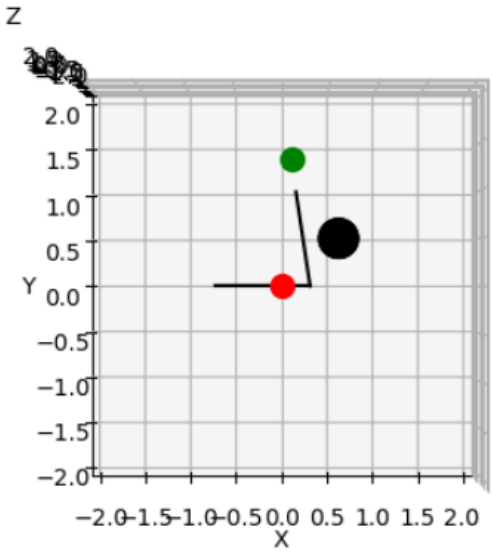
5. Try increasing the radius of the obstacle r. What do you observe? Also experiment with different starting configurations $\mathbf{q_0}$. Add a figure to the PDF for at least one increased obstacle radius, and add another figure for at least one different starting configuration. Discuss the results in your PDF answers.

```
Initial SSE Objective: 1.8667613974260215
Final SSE Objective: 0.5294775745419773
Solution
x1 = -0.34287223326259414
x2 = 1.0442452196919831
x3 = 1.5208920555754741
```



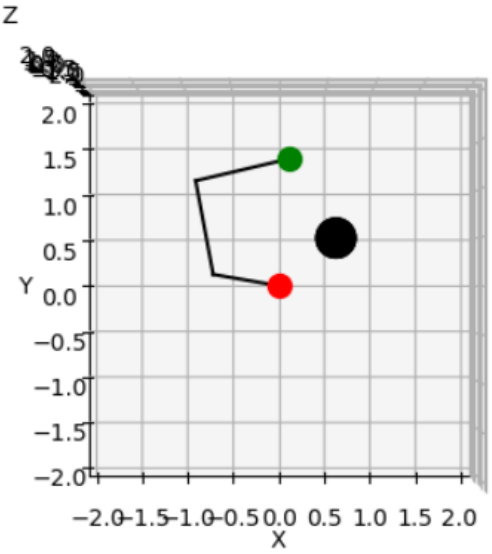
radius r = 0.6.
We can see that r increases, and robot arm now can't reach $\mathbf{q_d}$ because it is trying to avoid touching obstacle.

```
Initial SSE Objective: 5.792133760559658
Final SSE Objective: 0.11899293840847219
Solution
x1 = 3.141592653589793
x2 = 3.141592653589793
x3 = 1.7200539877392438
```



$\mathbf{q_0}$ = [3, 3, 3].
We observe that the robot arm cannot reach $\mathbf{q_d}$ this time. This could be due to the bounds restricting the robot arm or the distance between the initial angle and the target position being too far. As a result, the robotic arm may not be able to reach the target position within the limited range of joint movements.

```
Initial SSE Objective: 2.197415446167069
Final SSE Objective: 4.393911441995233e-07
Solution
x1 = 2.971845070071528
x2 = -1.2159722489025522
x3 = -1.5255727500320317
```



$\mathbf{q_0}$ = [1.8, 0, 0].
We can observe that the robot arm successfully reaches $\mathbf{q_d}$ this time by taking an alternative path that avoids the obstacle.