2.12: Introduction to Robotics
Lab 2:
Planar Kinematics*

Spring 2022

Assigned on: 10th February 2022 Due by: 17th February 2022

Instructions:

1. **You will be working on this lab with a partner during your lab section. Please make sure to attend!**
2. **Time permitting, a TA will verbally run through the questions attached to this handout during the lab section. Please think about the questions as you go through the lab. If we are constrained for time, we may ask you to upload your answers to Canvas.**

1 Introduction

In this lab, you'll be implementing the forward and inverse kinematics you learned for simple, planar articulated robots. You can find the required code for this lab at the class' GitHub: `https://github.com/mit212/lab2_2022`.

2 Forward Kinematics for a 2-link Planar Arm

2.1 The Workspace

As discussed in lecture, the region that a robot arm is able to reach is known as its **workspace**. The workspace can be represented in terms of both the robot's valid joint coordinates, or as the set of task coordinates reachable by the end effector. At the low-level, the commands sent to the actuators controlling each of the robot's joints must be in terms of the joint coordinates, but often as a roboticist you'll care about the task coordinates. In order to convert the valid joint coordinates for a robot to a reachable set of task coordinates (i.e. the workspace), we use forward kinematics. For a planar arm, the forward kinematics are straight forward to determine geometrically. For the

*

1. Version 1 - 2020: Jerry Ng, Rachel Hoffman, Steven Yeung, Kamal Youcef-Toumi
2. Version 1 - 2021: Cormac O'Neill
3. Version 2 - 2022: Cormac O'Neill

particular case of a 2-link, planar arm (fig. 1), with link lengths L_1 and L_2, and with joint angles q_1 and q_2, we use (Eq. 1) and (Eq. 2).

$$x = L_1 \cos q_1 + L_2 \cos (q_1 + q_2) \tag{1}$$

$$y = L_1 \sin q_1 + L_2 \sin (q_1 + q_2) \tag{2}$$

We are going to use this knowledge to calculate the workspace of a 2d planar robot in Matlab. To do so, open the provided file `FowardKinematics_Lab2.mlx` and go through it, and add the forward kinematic equations in lines 14 and 15. Make sure to answer the questions within. These questions have been repeated here for completeness.

Question 1 *The lengths of the robot's links will obviously constrain the outer edge robot's workspace, since it would be impossible for it to reach further than the length of both links combined in any one direction. But can the lengths of a robot's links also limit the lower limit of its workspace? i.e. do certain link lengths prevent the robot endpoint from reaching all the way back to its base?*

Question 2 *How is the robot's workspace limited? Are there certain regions that it will have difficulty reaching? Can you think of any reasons why this might be the case?*

Question 3 *Modify the code such that each joint is able to reach a range of angles from* 0° *to* 360°*. Experiment with the following cases: the robot links are of equal length (1 and 1), link 1 is longer than link 2 (1 and 0.5), link 1 is shorter than link 2 (0.5 and 1). How does this affect the reachable workspace?*

3 Inverse Kinematics for a 2-link Planar Arm

3.1 Inverse Kinematics

In this section, you will be operating on `TwoDInverseKinematics_Student.m` and inputting the inverse kinematics into the function `ik_student.m`.

Two-link manipulator The two-link manipulator is shown in Figure 1, in which we define the length of two links (L_1, L_2) in meters, and joint positions (q_1, q_2) in rad, and target endpoint position (x, y). **Both q_1 and q_2 are defined as positive about the Z axis (out of the page).** The inverse kinematic solution follows that of fig. [1]:

$$q_2 = \pm 2 \cdot \text{atan2}(\sqrt{(L_1 + L_2)^2 - (x^2 + y^2)}, \sqrt{(x^2 + y^2) - (L_1 - L_2)^2}), \tag{3}$$

and

$$q_1 = \text{atan2}(y, x) - \text{atan2}(L_2 \sin q_2, L_1 + L_2 \cos q_2), \tag{4}$$

where atan2(y,x) is a function that computes the arc tangent two variables y and x. It uses the signs of both arguments to determine the correct quadrant of the result. Note that there are two solutions for (q_1, q_2) that corresponds to elbow-up and elbow-down configurations.

Controlling robot with joint position is the most basic strategy. However, we want the endpoint of the robot arm go to a designated position. That is the inverse kinematics that we will complete.

Fill in lines 27 and 28 in function `ik_student(x,y,q0,L1,L2, true)`, used in

`TwoDInverseKinematics_student.m` on line 83. Use equations (3), and (4) to do this. You can find the contents of this function in the `ik_student.m` file. The L_1 and L_2 values that will be tested are defined within `TwoDInverseKinematics_student.m`, but your code should work for an

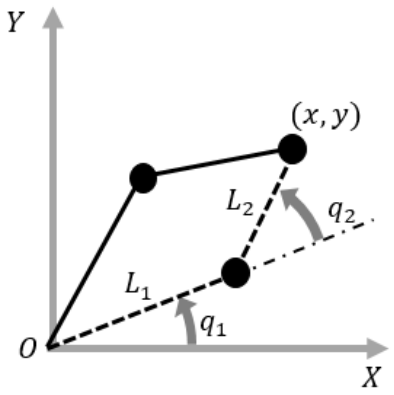

Figure 1: The kinematics of a two-link manipulator. Both the forward and inverse kinematics can be found in the lecture notes.

arbitrary, feasible pair of link lengths. Note that the function has an input argument `q0` describing the current joint positions as a Matlab array: `[q1, q2]`. The purpose of feeding the correct position is that when there are multiple solutions, we will choose the solution that is closest to the current position.

3.2 Trajectory Planning

We wish to simulate a robot arm moving in a circle with a given radius and center. To do so, you're going to run the file `TwoDInverseKinematics_student.m` with your own inverse kinematics. There will be comments in the code specifically indicating where to add your function. Run the script for the provided circular trajectory, centered at $(0.3, 0.1)$ and with a radius of 0.15.

Question 4 *Notice that the function for* `ik_student` *calculates two possible pairs of joint coordinates. Why is that done? How does it select the best candidate? How would this change as the number of links increases/decreases? See what happens when we don't enforce the correct solution by changing the flag in* `ik_student(x,y,q0,L1,L2, true)` *from true to false.*

Question 5 *Would a real robot arm actually yield the commanded trajectory? Why/why not? Hint: Consider what might differ between a simulation and a real piece of hardware.*

References

[1] D. Gordon. Robotics: Forward and inverse kinematics. [Online]. Available: http://www.slideshare.net/DamianGordon1/forward-kinematics

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[3] H. Asada and J.-J. E. Slotine, *Robot analysis and control*. John Wiley and Sons, 1986.