2.12: Introduction to Robotics
Lab 4: Motion of Robot Arms*

Spring 2023

Assigned on: 2nd March 2023. Due by: 8th March 2023.

Instructions:

1. **When you are done with the lab, call a lab staff to do your check-off.**
2. **(Optional) Upload a video showing the robot arm in motion.**

1 Introduction

In this lab, you will go through the motion of robot arms, including:

1. Using forward kinematics (FK) to find the endpoint position given joint positions.
2. Using inverse kinematics (IK) to find joint positions given some target endpoint position (Sometimes called the "tool center point" (TCP)).
3. Scripting a complex trajectory.

NOTE: Set your robot arm back to the home position before each run.

2 Setting up the code

Please make sure your robotic arm is assembled (replace the disk with the 2nd link) and wired correctly (see assembly instructions from Lab 4), and then download Arduino code from Canvas or Github.

```
cd ~                          # make sure we are at home folder
git clone https://github.com/mit212/lab4_2023.git
```

*

1. Version 1 - 2016: Peter Yu, Ryan Fish and Kamal Youcef-Toumi
2. Version 2 - 2017: Yingnan Cui, Kamal Youcef-Toumi, Steven Yeung and Abbas Shikari
3. Version 3 - 2019: Daniel J. Gonzalez
4. Version 4 - 2020: Jerry Ng, Steven Yeung, Rachel Hoffman, Kamal Youcef-Toumi
5. Version 5 - 2021: Hanjun Song
6. Version 6 - 2023: Ravi Tejwani and Kentaro Barhydt

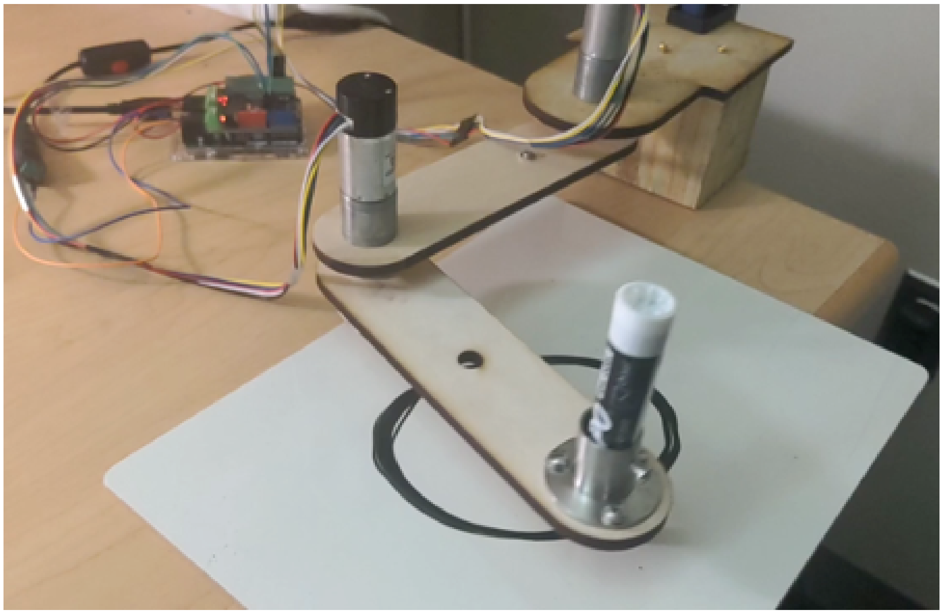

Figure 1: 2.12/2.120 two-link manipulator.

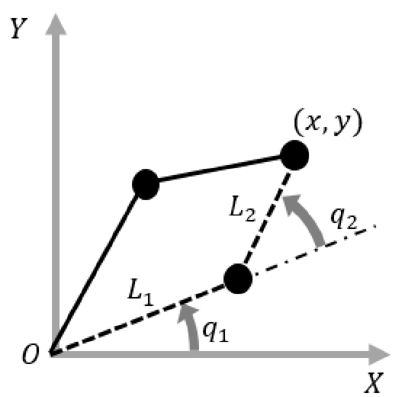

Figure 2: The kinematics of two-link manipulator. **Note that in the configuration shown, q_1 and q_2 have positive values.**

2.1 Forward Kinematics

In this section, you will be implementing the forward kinematics (FK) of the two-link manipulator. The two-link manipulator is shown in Figure 2, 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) in meters. **Both q_1 and q_2 are defined as positive about the Z axis (out of the page).** The goal is to compute the endpoint position when the joint positions are given and we will compare the computed (true) endpoint position with the actual endpoint position.

Find the forward kinematics block in the main loop of the Arduino code and implement it following the instructions in the code. Once you are done, upload the code to the Arduino and see if the robot moves to the expected endpoint position. You can see the actual endpoint position through the Serial Monitor. The forward kinematics equations are given below as we learned from Lab 2.

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

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

Question 1 *How are the true endpoint position and actual endpoint position different? You can use a ruler to measure the actual endpoint position. Where does the difference come from?*

2.2 Inverse Kinematics

In this section, you will be implementing the inverse kinematics of the two-link manipulator. The goal is to compute the joint positions when a target endpoint position is given. We will compare the target endpoint position with the actual endpoint position. Find the inverse kinematics block in the main loop of the Arduino code and implement it following the instructions in the code. Once you are done, upload the code to the Arduino and see if the robot moves to the target endpoint position. As we learned in Lab 2, the inverse kinematic solution follows that of [1]:

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

and

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

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. You may want to use the following functions in the Arduino code.

- `atan2()`, `cos()`, `sin()`, `sqrt()`
- `PI`: constant π
- `sq(x)`: x^2

Question 2 *How are the target endpoint position and actual endpoint position different? Where does the difference come from?*

Question 3 *Can you see the difference between the elbow-up and elbow-down solutions? Please take a picture of each solution.*

Question 4 *What safety features are put in place to make sure that the robot can only be commanded to a position within the workspace?*

2.3 Trajectory Planning

In this section, you will make the robot arm move in a circle with a given radius and center. To do so, you are going to implement the circular path block in the main loop of the Arduino code. There will be comments in the code specifically indicating variables that need to be changed and assigned. The equation of a circle in polar coordinate will be useful to generate the circular path.

Fine-tune the following parameters to improve your circle:

- Controller gains for Motor 1 and Motor 2.
- Sampling period (variable name: `period_us`).
- Step size of waypoints (change the line `i++;` to `i += 2;` or some other integer increment).

In addition to looking at the circle drawn by the robot arm, you can use the Matlab script "SerialRead2.m" to capture and analyze the data (see Fig. (3) so as to identify the issues. Make sure to uncomment `#define MATLAB_SERIAL_READ` and comment out `#define PRINT_DATA` before uploading the code to Arduino for this to work. Also make sure to change the serial port name in the Matlab script to match the Arduino port before running the script.

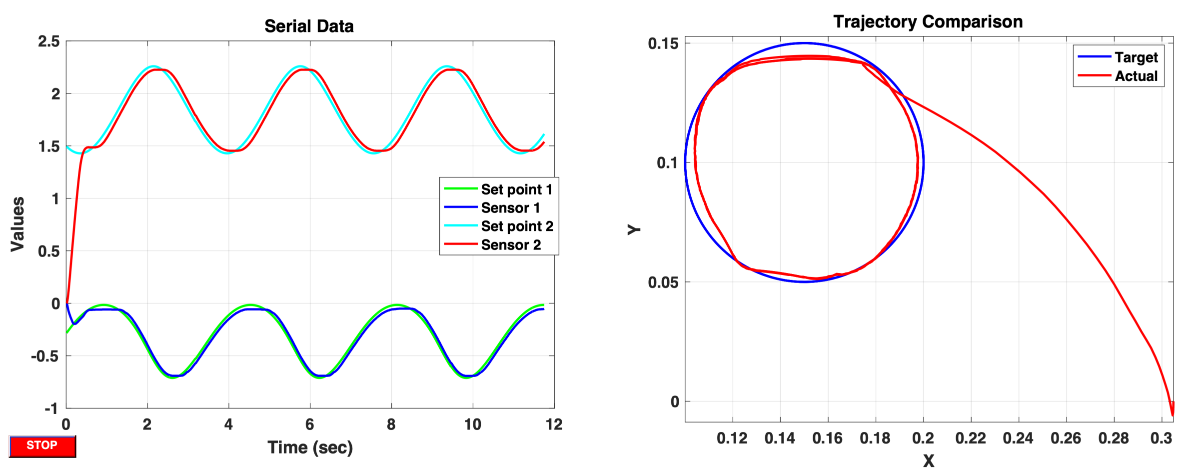

Figure 3: SerialRead.m outputs.

(Optional) Upload a video showing the robot arm in motion to this Dropbox.

If time permits, program a different trajectory for the robot to follow.

Question 5 *Does the robot arm actually yield the commanded trajectory? Why/why not? Show a picture of the best circular path that your robot arm drew on the white board.*

Question 6 *(Optional) Can you make the robot follow a different shape of path? Show a picture of the path that your robot arm drew on the white board.*

References

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