ID6040: Introduction to Robotics

Ayush Mukund Jamdar EE20B018

March 5, 2023

1 Introduction to Assignment 1

The assignment aims to let the learner implement a basic **Forward Kinematics** solution to an arm with two revolute joints to get the end-effector coordinates. Furthermore, the assignment scripts together are an introduction to interfacing **CoppeliaSim (Edu)** with Python3.

2 The Assignment

2.1 The Problem

The module `fw_kin.py` implements the forward kinematics function `fw_kin()`. It takes one argument - `joint_angles` - a Python list with angles of the two arms, in degrees. The function returns the x and y coordinates of the end-effector in a list. In the programming assignment, I write code for this function. Everything else - connecting to CoppeliaSim, the design, simulation interface and the testbench - is already implemented.

2.2 Forward Kinematics

The given system is represented in Figure 1. Let l_1 be the length of the arm from the origin and l_2 the length of the next one. It is easy to see that the x and y coordinates of the endpoint are given by -

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

2.3 A Python Function

The above equations are implemented in the Python module `fw_kin.py` as follows.

```
def fw_kin(joint_angles):
# Compute end effector position [in meters] for the given pair of joint angles [in degrees]

# convert angles to radians (Note that 3.14 is used instead of math.pi)
joint_angles_rad = [angle * (3.14 / 180) for angle in joint_angles]
```

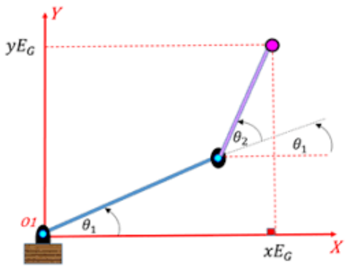
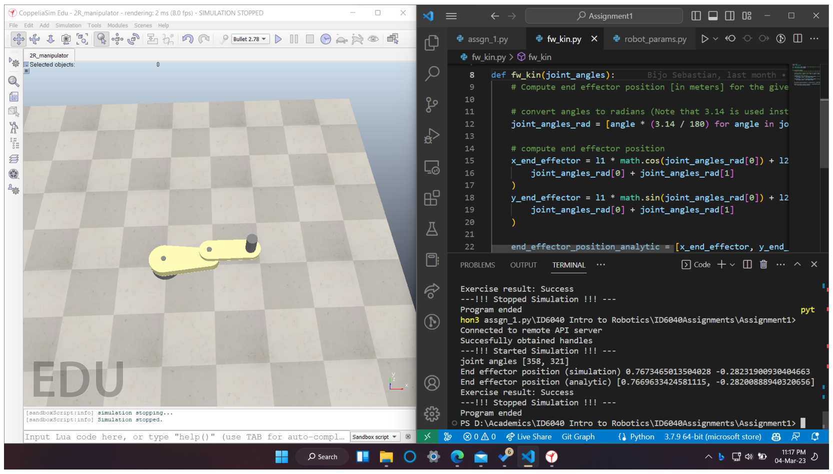


Figure 1: A graphical representation of the system.

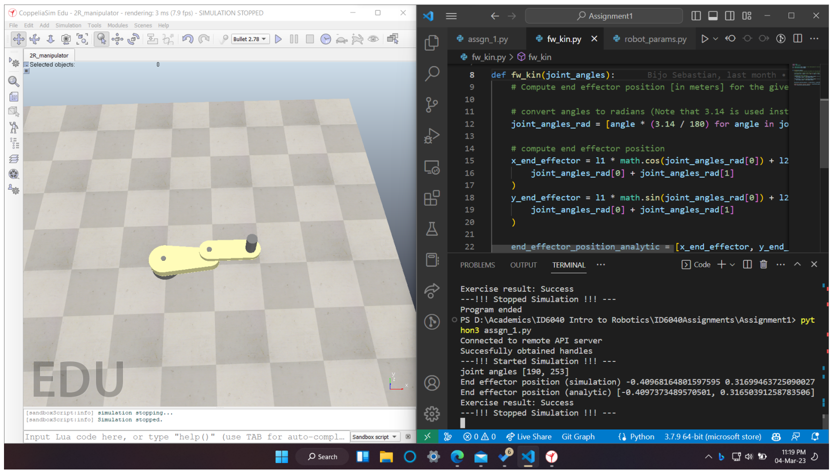


```
def fw_kin(joint_angles):        Bijo Sebastian, last month •
    # Compute end effector position [in meters] for the give

    # convert angles to radians (Note that 3.14 is used inst
    joint_angles_rad = [angle * (3.14 / 180) for angle in jo

    # compute end effector position
    x_end_effector = l1 * math.cos(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )
    y_end_effector = l1 * math.sin(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )

    end_effector_position_analytic = [x_end_effector, y_end_
```

```
[sandboxScript:info] simulation stopping...
[sandboxScript:info] Simulation stopped.
```

```
Exercise result: Success
---!!! Stopped Simulation !!! ---
Program ended                                                          pyt
hon3 assgn_1.py\ID6040 Intro to Robotics\ID6040Assignments\Assignment1>
Connected to remote API server
Succesfully obtained handles
---!!! Started Simulation !!! ---
joint angles [358, 321]
End effector position (simulation) 0.7673465013504028 -0.28231900930404663
End effector position (analytic) [0.7669633424581115, -0.28200888940320656]
Exercise result: Success
---!!! Stopped Simulation !!! ---
Program ended
PS D:\Academics\ID6040 Intro to Robotics\ID6040Assignments\Assignment1>
```

Figure 2: Run-1



```
def fw_kin(joint_angles):        Bijo Sebastian, last month •
    # Compute end effector position [in meters] for the give

    # convert angles to radians (Note that 3.14 is used inst
    joint_angles_rad = [angle * (3.14 / 180) for angle in jo

    # compute end effector position
    x_end_effector = l1 * math.cos(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )
    y_end_effector = l1 * math.sin(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )

    end_effector_position_analytic = [x_end_effector, y_end_
```

```
[sandboxScript:info] simulation stopping...
[sandboxScript:info] Simulation stopped.
```

```
Exercise result: Success
---!!! Stopped Simulation !!! ---
Program ended
PS D:\Academics\ID6040 Intro to Robotics\ID6040Assignments\Assignment1> pyt
hon3 assgn_1.py
Connected to remote API server
Succesfully obtained handles
---!!! Started Simulation !!! ---
joint angles [190, 253]
End effector position (simulation) -0.40968164801597595 0.31699463725090027
End effector position (analytic) [-0.4097373489570501, 0.31650391258783506]
Exercise result: Success
---!!! Stopped Simulation !!! ---
```

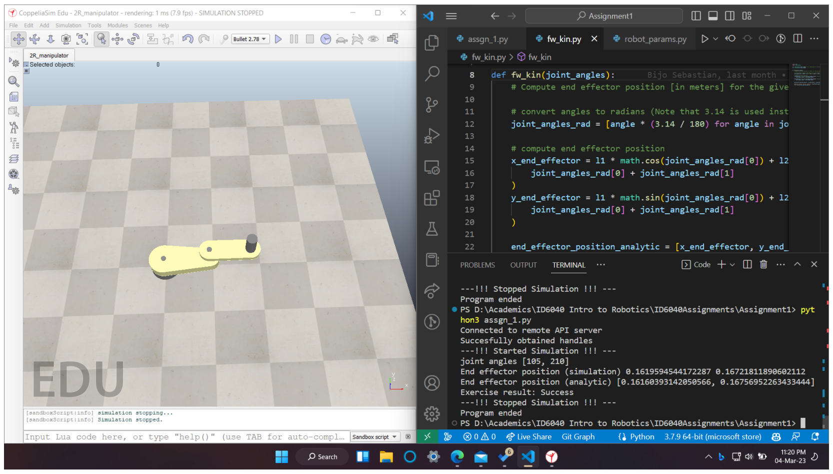
Figure 3: Run-2

```
# compute end effector position
x_end_effector = l1 * math.cos(joint_angles_rad[0]) + l2 * math.cos(
    joint_angles_rad[0] + joint_angles_rad[1]
)
y_end_effector = l1 * math.sin(joint_angles_rad[0]) + l2 * math.sin(
    joint_angles_rad[0] + joint_angles_rad[1]
)

end_effector_position_analytic = [x_end_effector, y_end_effector]

return end_effector_position_analytic
```

The comments describe the action of every line. Please note that due to an inconsistency in the value of π in the given assignment folder, 3.14 is used instead.

2.4 Simulation Results

Figures 2, 3 and 4 are three successful runs of the simulation with random joint angles. Please note that the screenshot was taken after the run due to which the scene object has returned to its default state and not the joint angles the sim started with. The points are printed out in the terminal.



```python
def fw_kin(joint_angles):
    # Compute end effector position [in meters] for the give

    # convert angles to radians (Note that 3.14 is used inst
    joint_angles_rad = [angle * (3.14 / 180) for angle in jo

    # compute end effector position
    x_end_effector = l1 * math.cos(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )
    y_end_effector = l1 * math.sin(joint_angles_rad[0]) + l2
        joint_angles_rad[0] + joint_angles_rad[1]
    )

    end_effector_position_analytic = [x_end_effector, y_end_
```

```
---!!! Stopped Simulation !!! ---
Program ended
PS D:\Academics\ID6040 Intro to Robotics\ID6040Assignments\Assignment1> pyt
hon3 assgn_1.py
Connected to remote API server
Succesfully obtained handles
---!!! Started Simulation !!! ---
joint angles [105, 210]
End effector position (simulation) 0.1619594544172287 0.1672181189060212
End effector position (analytic) [0.16160393142050566, 0.16756952263433444]
Exercise result: Success
---!!! Stopped Simulation !!! ---
Program ended
PS D:\Academics\ID6040 Intro to Robotics\ID6040Assignments\Assignment1>
```

CoppeliaSim Edu - 2R_manipulator - rendering: 1 ms (7.9 fps) - SIMULATION STOPPED
Bullet 2.78
2R_manipulator
Selected objects
EDU
[sandboxScript:info] simulation stopping...
[sandboxScript:info] Simulation stopped.
Input Lua code here, or type "help()" (use TAB for auto-compl...
Sandbox script
Assignment1
assgn_1.py
fw_kin.py
robot_params.py
PROBLEMS
OUTPUT
TERMINAL
Live Share
Git Graph
Python
3.7.9 64-bit (microsoft store)
11:20 PM
04-Mar-23

Figure 4: Run-3

2.5 Conclusion

More than implementing a simple forward kinematics equation, the assignment has been about an introduction to CoppeliaSim. I got a brief overview of the three rudimentary blocks of the software - scene objects, calculation modes, and control mechanisms. This simulation was performed on the Bullet 2.78 physics engine.