Lab 1 report

Jiahui Li

October 10, 2016

1 Introduction

In my environment, I rarely find a physical kinematic linkage that has 4 joints. But I find some foldable lamp have three joints and if we rotate the lamp, it will have one more joint. The picture is shown below.

Figure 1: Physical kinematic linkage.

Here I outline the design of this linkage in terms of its joint space. The picture is shown below. From the picture, we can see all of this four joints are revolute joints.

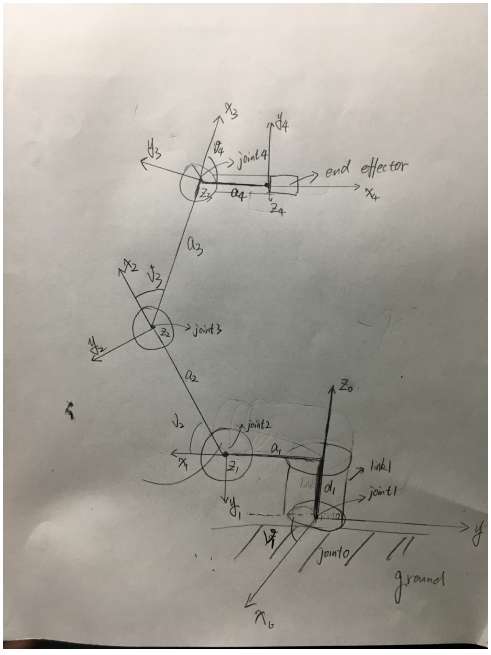

Figure 2: Outline of the linkage

The intended functionality of this linkage is to set the lamplight in a good position when people study, and people can carry it easily because of this foldable structure. As it is shown in the figure1, the lamp arm is really nice.

In order to analyze the it, we choose a linear motion between two points, more detail will be discussed in the results part.

2 Methods

In this section, I will compute the forward kinematics and implement inverse kinematics.

2.1 Forward kinematics

Since I just see it on the Amazon, I don't have the exact parameter values of this lamp. But we can define the Denavit-Hartenberg parameters as below:

- α : angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise
- a : distance between O_i and O_{i^1}
- θ : angle between axes x_{i-1} and x_{i-i} about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.
- d : coordinate of O_{i^1} along z_{i-1}

Based on the figure 2, we can determine the Denavit-Hartenberg parameters for the linkage like this and I give these constant based on my experience, where θ is the variables.

Link	a_i	α_i	d_i	θ_i
1	5cm	-90	1	v_1
2	15cm	0	0	v_2
3	15cm	0	0	v_3
4	10cm	0	0	v_4

Table 1: The parameters of Denavit-Hartenberg

I use python to create this program. Please see it in attachment.

2.2 Inverse kinematics

In this part, I will use inverse kinematics to get the joint variables by given the position of end effector. The method I choose is to use pesudo-inverse method. Below is a figure of how this method works and more details can be seen in the code file.

```
while (e is too far from g){
    compute the Jacobian matrix J
    compute the pseudoinverse of the Jacobian matrix— J+
    compute change in joint DOFs: Δθ = J+ · Δe
    apply the change to DOFs, move a small step of αΔθ: θ = θ + αΔθ
}
```

Figure 3: Flow chart of inverse kinematic.

3 Results

In this section, I will present and analyze the results of your computation/experiments. In order to compare to the first problem, I tried two different points in the operational space. The position and the specified D-H parameters are shown below.

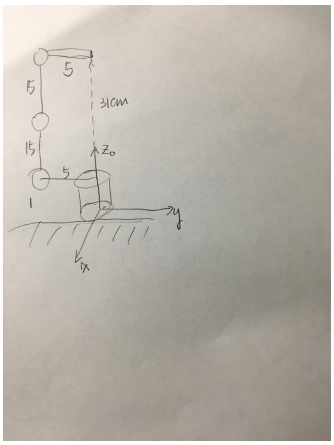

Figure 4: Position 1

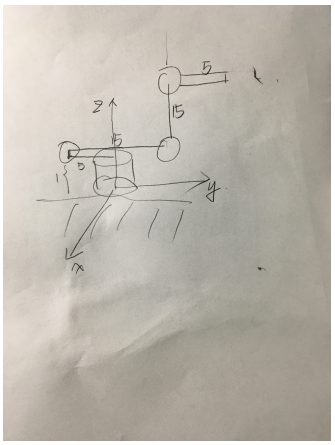

Figure 5: Position 2

Link	a_i	α_i	d_i	θ_i
1	5cm	-90	1	-90
2	15cm	0	0	-90
3	15cm	0	0	0
4	10cm	0	0	-90

Table 2: The parameters of Denavit-Hartenberg in case 1

Link	a_i	α_i	d_i	θ_i
1	5cm	-90	1	-90
2	15cm	0	0	-180
3	15cm	0	0	90
4	10cm	0	0	-90

Table 3: The parameters of Denavit-Hartenberg in case 2

In situation 1, the value of z should be 31cm and x =0, y=0. In situation 2, z should be 16 and y =15, x = 0. The program results are shown below. We can see that the results are pretty the same.

```
[[ -0.  -0.   1.  -0.]
 [  1.  -0.   0.  -0.]
 [  0.   1.   0.  31.]
 [  0.   0.   0.   1.]]
```

Figure 6: Program result of Position 1

```
....
[[ -0.  -0.   1.  -0.]
 [  1.  -0.   0.  15.]
 [  0.   1.   0.  16.]
 [  0.   0.   0.   1.]]
```

Figure 7: Program result of Position 2

And then I would like to plot the trajectory of this linkage. First I define that the range of joint0 is[-180,180], the range of joint1 is [-180,0], the range of joint2 is [0,180], and the joint3 is [-180,0]. Using the method in part2, we can get the 3D image below. But it can not represent the trajectory of the linkage very well. We can fixed the joint1, in this way, it will become a 2D image, since the linkage only moves in the zoy plane. The 2D image is also shown below.

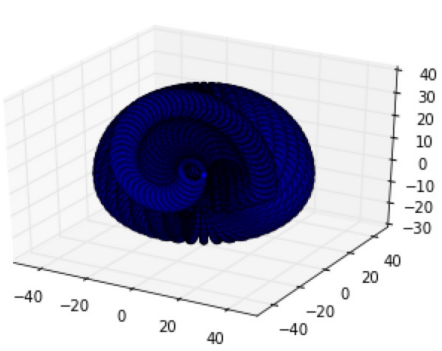

Figure 8: Program result of Position 1

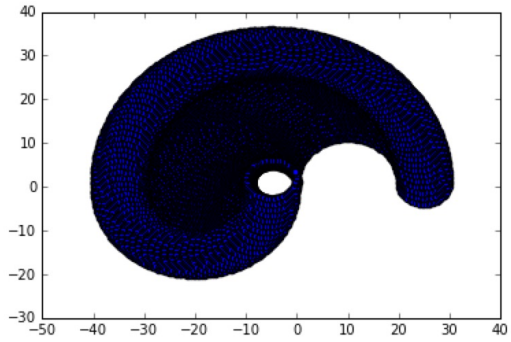

Figure 9: Program result of Position 2

Then I use IK method to compute the configuration parameters of the endpoints of the desired motion. And use FK to generate a trajectory in operational space arising from a linear interpolation between these in configuration space. The trajectory is shown below. As we can see from the picture, It doesn't work as well as I hope. So I am still working on it.

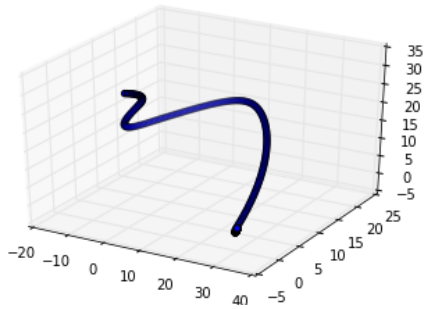

Figure 10: Trajectory of the end effector

Finally, it takes me around two days to finish this lab. Most of time, I'm figuring out why I am wrong.